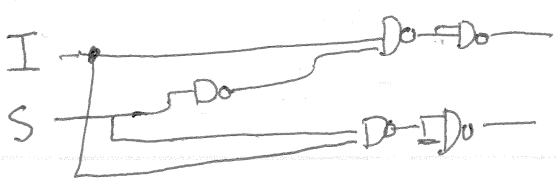
4-3/ ~~Decoder~~ Demux /



If I is always TRUE, Decoder

S	A	B
0	1	0
1	0	1

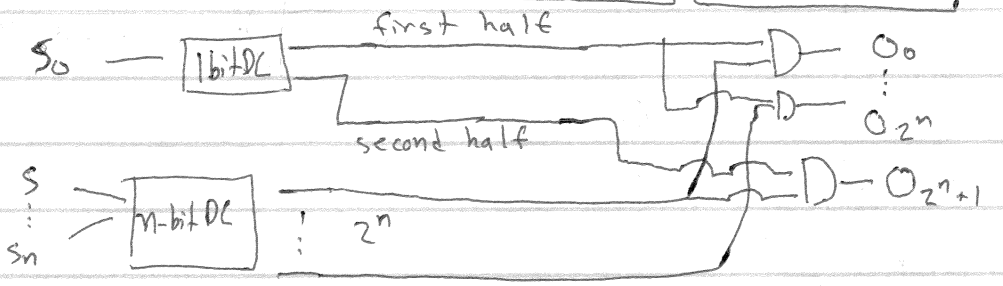
S	B	A
0	0	1
1	1	0

2-bit decoder ⇒ 4 output wires

0 0	0 0 0 1
0 1	0 0 1 0
1 0	0 1 0 0
1 1	1 0 0 0

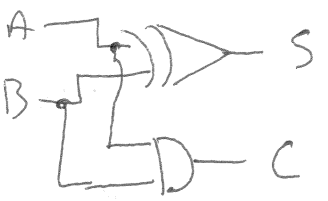
n+1 -bit decoder ⇒ 2^{n+1} output wires

S_0 S_1 ... S_n | O_0 ... O_{2^n} | O_{2^n+1} ... $O_{2^n+2^n}$



1-4/

A	B	C	S
0	0	0	0
1	0	0	1
0	1	0	1
1	1	1	0

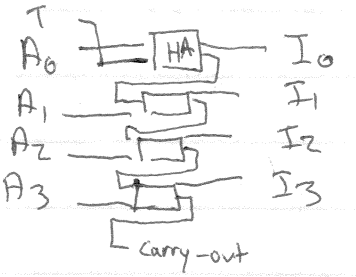


Half-Adder

2-bit increment

$HA(A_0, T) = (C_0, S_0)$

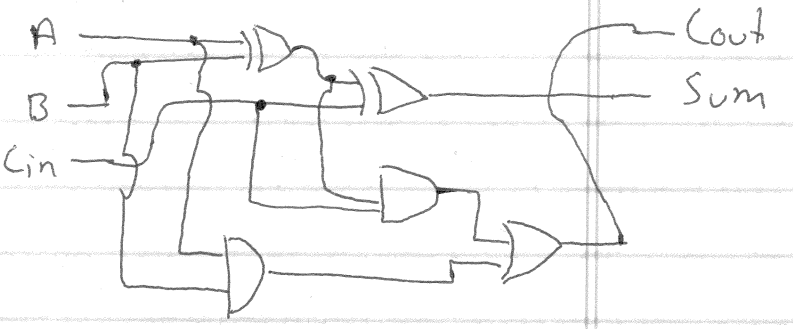
$HA(A_1, C_0) = (C_1, S_1)$

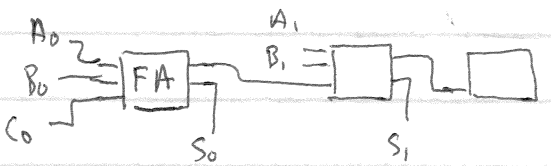


Next time:
Full Adder
ALU

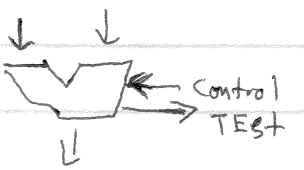
-1/

A	B	Cin	Cout	Sum
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





ALU - Arithmetic Logic Unit



ALU should be able to compute this:

ALU (A, B, C) = Ans

Test = (ans=0?, ans<0)

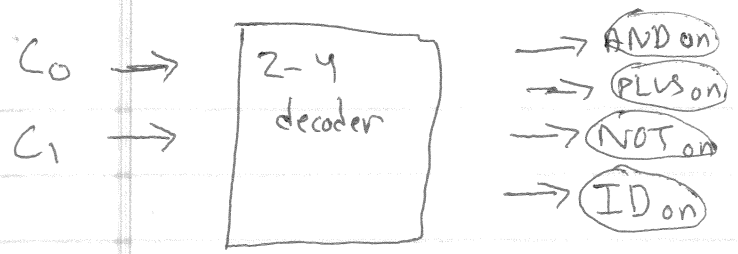
Ans:
- A B
- A + B
- $\bar{A}$
- A

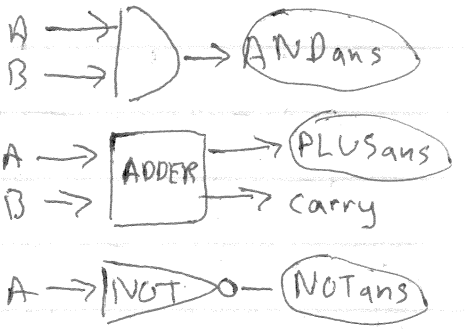
if C = AND
C = PLUS
C = NOT
C = A / ID

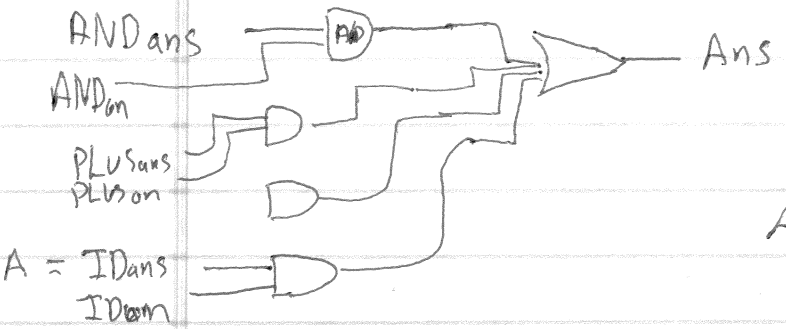
5-2

C = 0, 1, 2, 3
00, 01, 10, 11

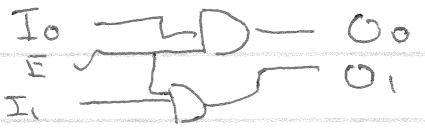
Input = (A), (B), (C)
Output = (Ans), (Test)





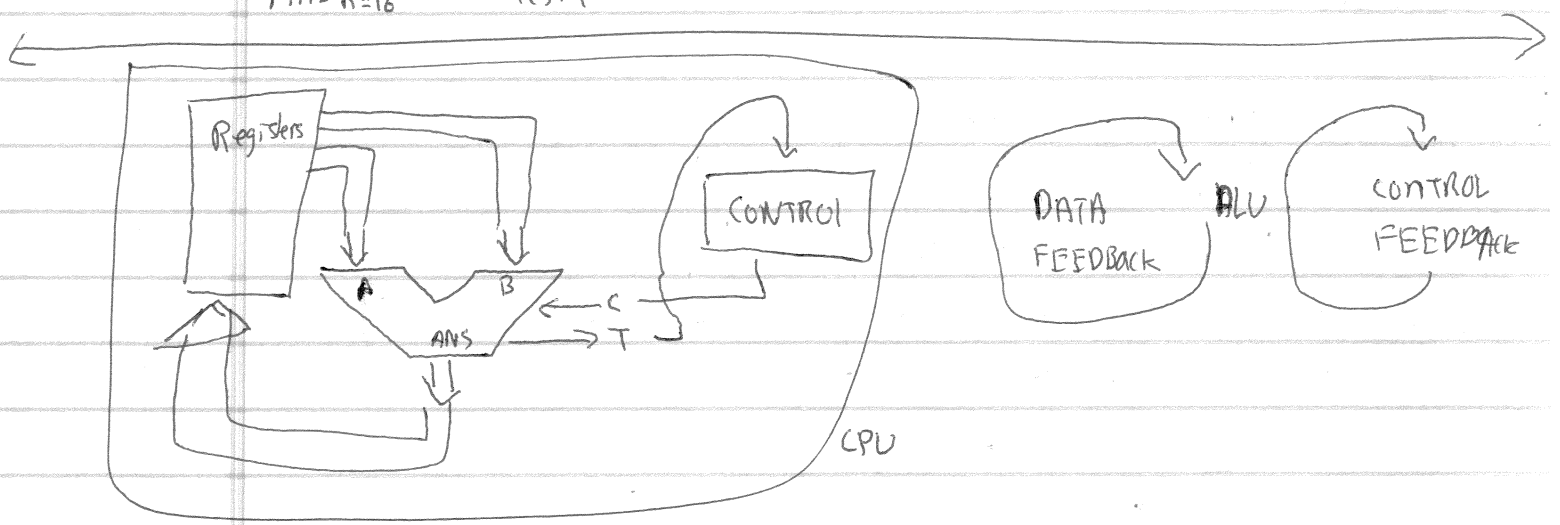


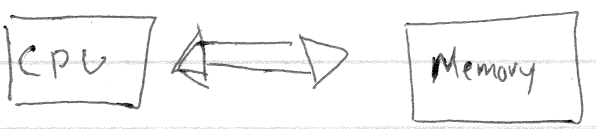
AND' (many wires, one wire) = many



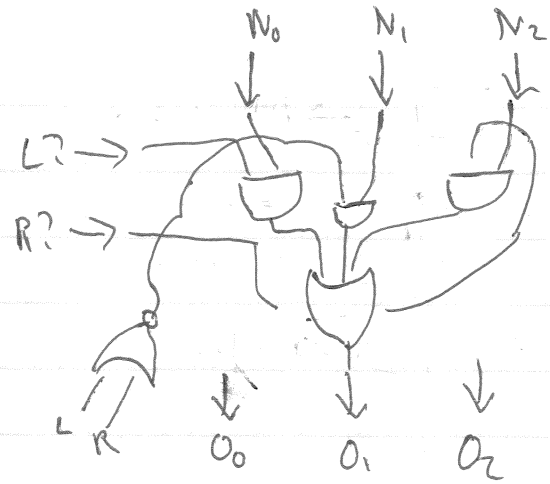
$Test_0 = Z?$

$Ans_{n=16} = Test_1 = N?$





5-3/ Shifter (Left?, Right?, N, O)



NOR = "Both False"

X	Y	
0	0	1
0	1	0
1	0	0
1	1	0

A

< ... , +, LS, +RP ... , PC, PC, +1, ... >

AMUX=A

PC := (PC+1) << 1 (and read)

AMUX=MBR PL := (PC+MBR) << 1

5-1/ Micro-sequence Logic

Input: N, Z, COND/2

Output: Mmux-control (0 = +1, 1 = ADDR)

COND = 00 – Never Jump 10 – Jump on Zero

01 – Jump on Neg 11 – Always Jump

N	Z	COND	Mmux c
———		00	0
———		11	1
1	—	01	1
0	—	01	0
—	1	10	1
—	0	10	0