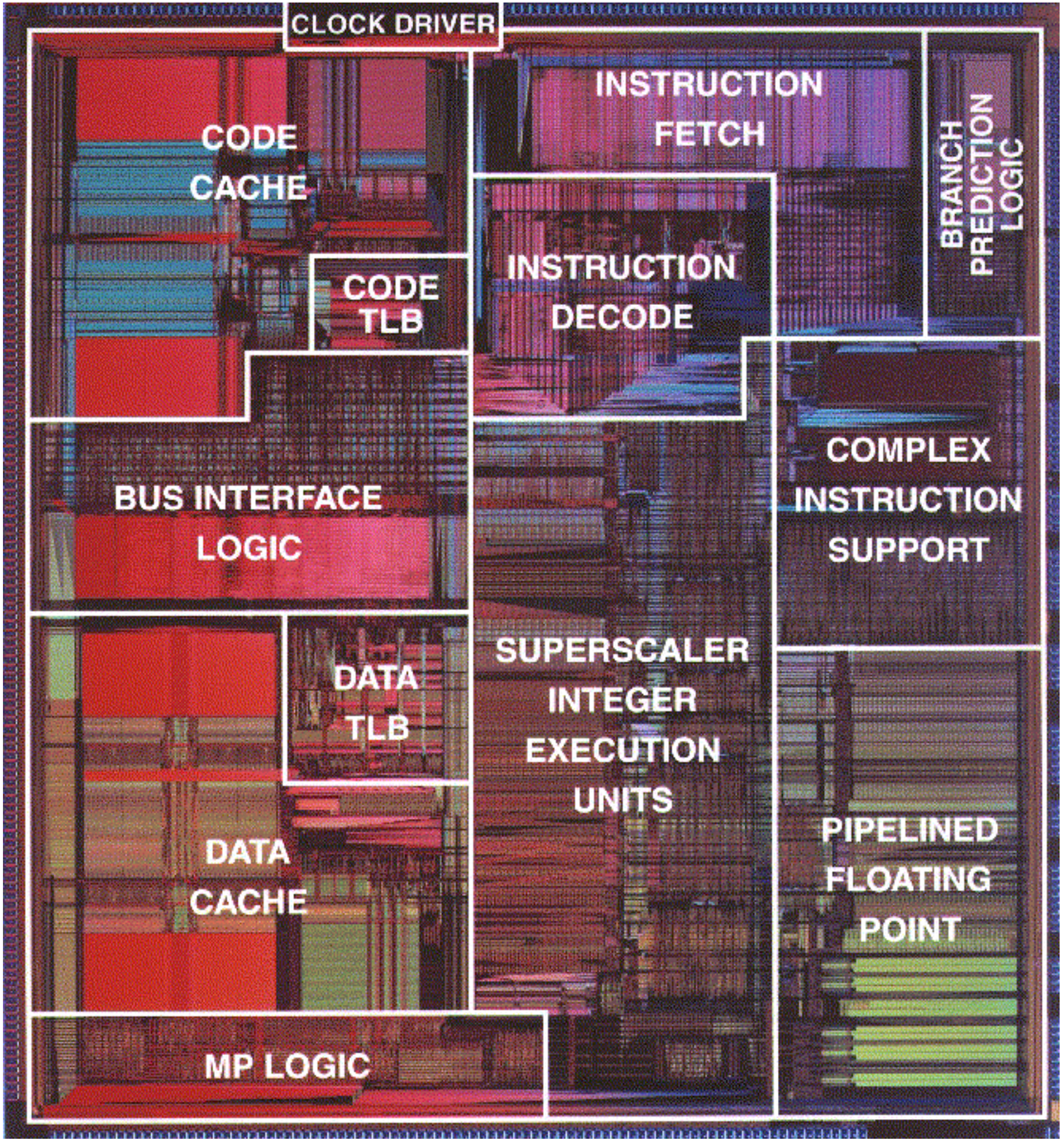
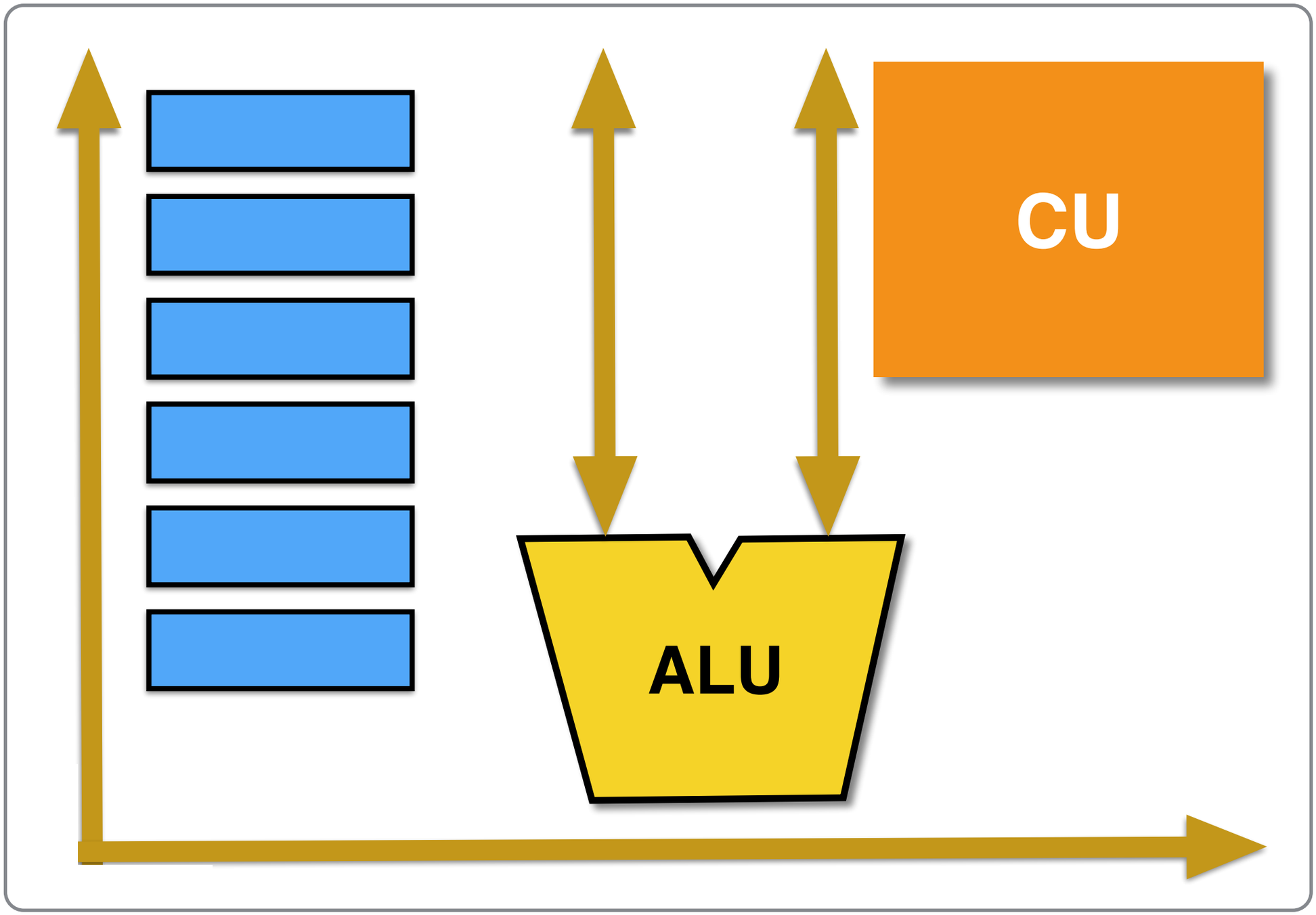
SMITH COLLEGE
COMPUTER SCIENCE

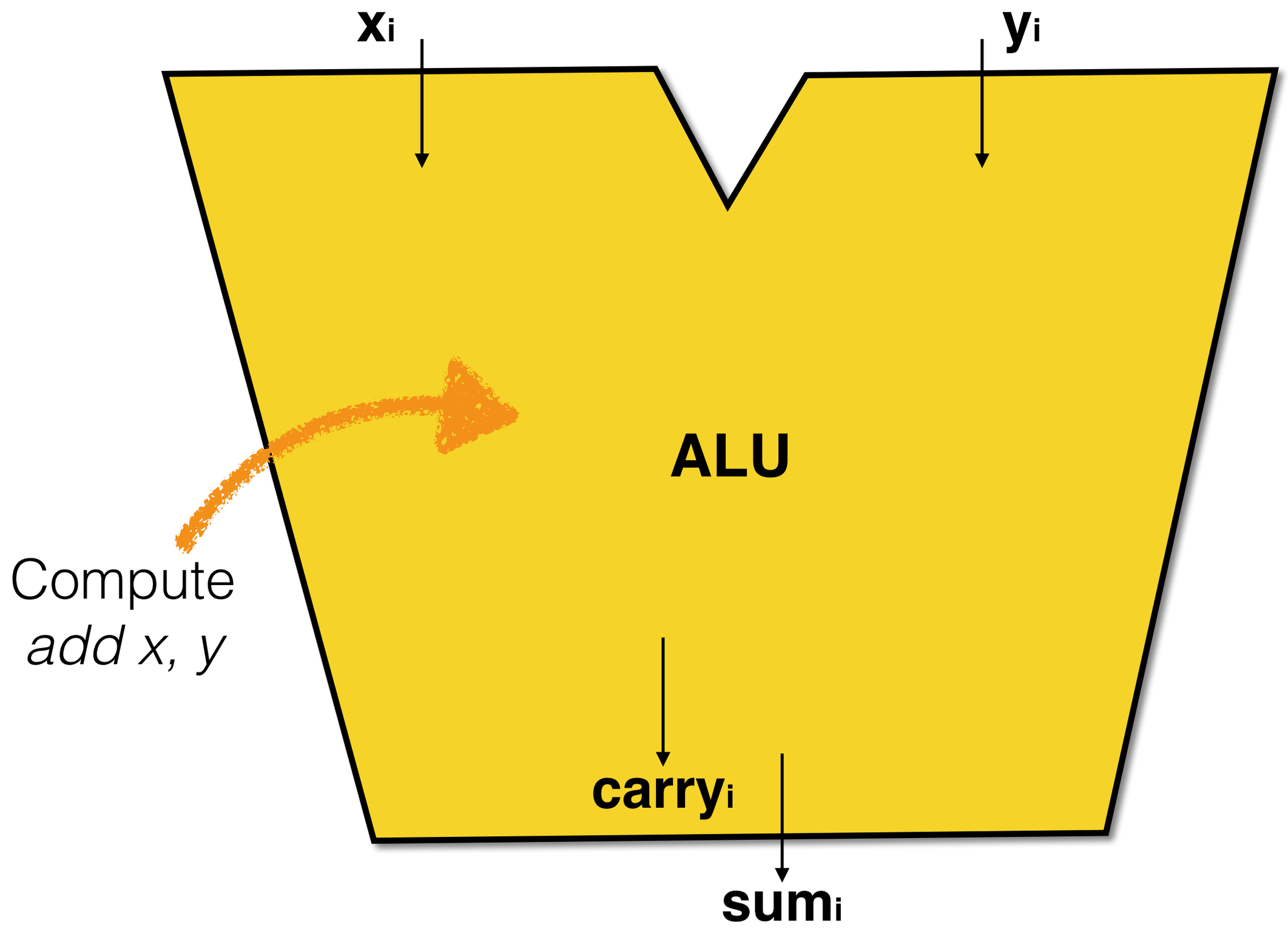
Number Systems

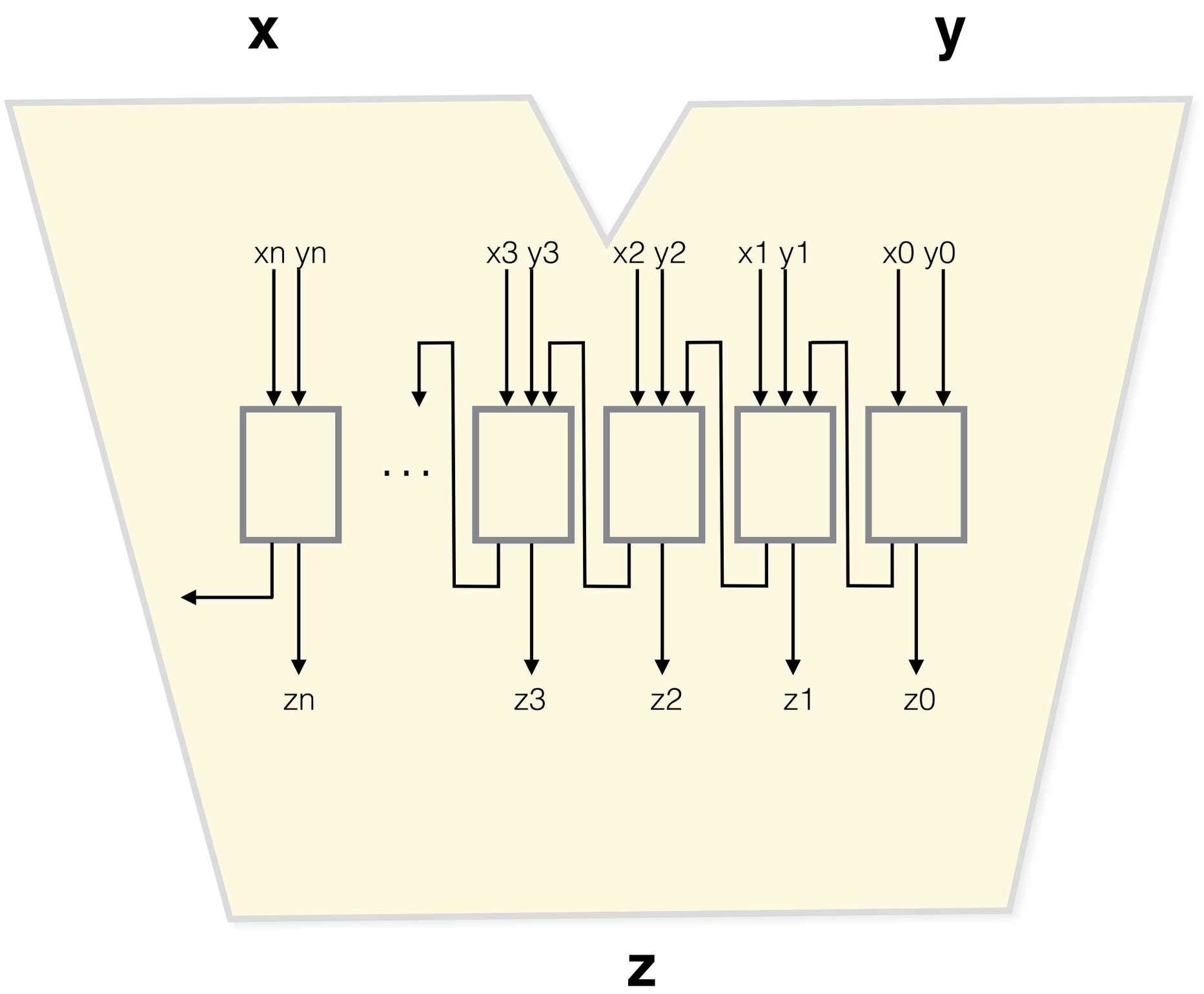
CSC231 — D. Thiebaut

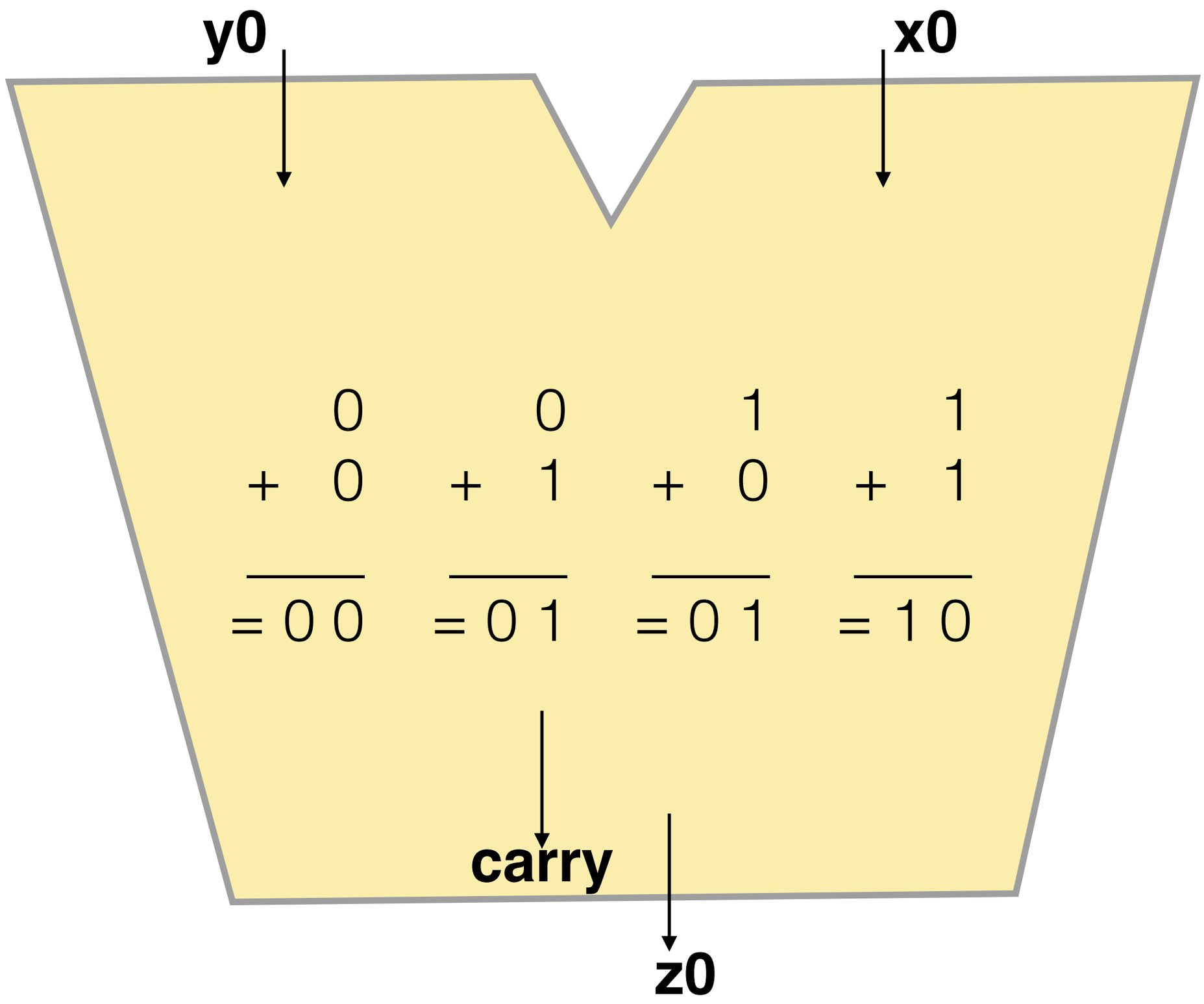

CLOCK DRIVER
CODE CACHE
INSTRUCTION FETCH
BRANCH PREDICTION LOGIC
CODE TLB
INSTRUCTION DECODE
BUS INTERFACE LOGIC
COMPLEX INSTRUCTION SUPPORT
DATA TLB
SUPERSCALER INTEGER EXECUTION UNITS
DATA CACHE
PIPELINED FLOATING POINT
MP LOGIC

Processor




x_i
y_i
ALU
Compute
add x, y
$carry_i$
sum_i


x
y
xn yn
x3 y3
x2 y2
x1 y1
x0 y0
zn
z3
z2
z1
z0
z

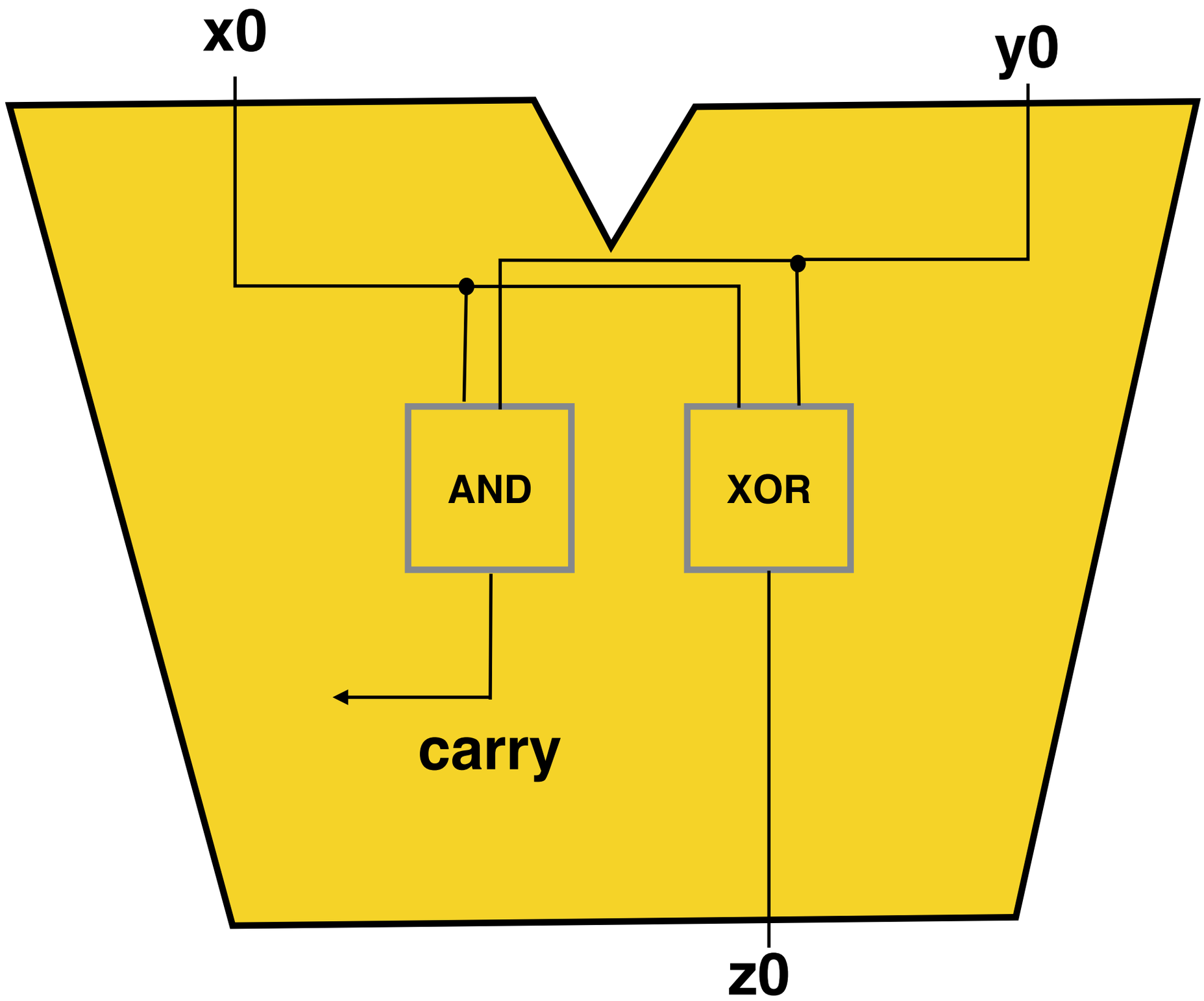

y0
x0
0 0 1 1
+ 0 + 1 + 0 + 1
= 0 0 = 0 1 = 0 1 = 1 0
carry
z0

x0	y0	Carry	z0
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

x0	y0	Carry	z0
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Carry = x0 **and** y0

z0 = x0 **xor** y0


x0
y0
AND
XOR
carry
z0

Moral of the Story:
Addition is performed by logic operations using *natural* binary numbers…

(*unsigned arithmetic*)

How can we represent signed binary numbers when all we have are **bits** (0/1)?

Whichever system we use should work with the binary adder in the ALU…

4-bit Nybble

Binary	Hex	Unsigned Decimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

4-bit Nybble

Sign Bit

Binary	Hex	Unsigned Decimal
0 000	0	0
0 001	1	1
0 010	2	2
0 011	3	3
0 100	4	4
0 101	5	5
0 110	6	6
0 111	7	7
1 000	8	8
1 001	9	9
1 010	A	10
1 011	B	11
1 100	C	12
1 101	D	13
1 110	E	14
1 111	F	15

4-bit Nybble

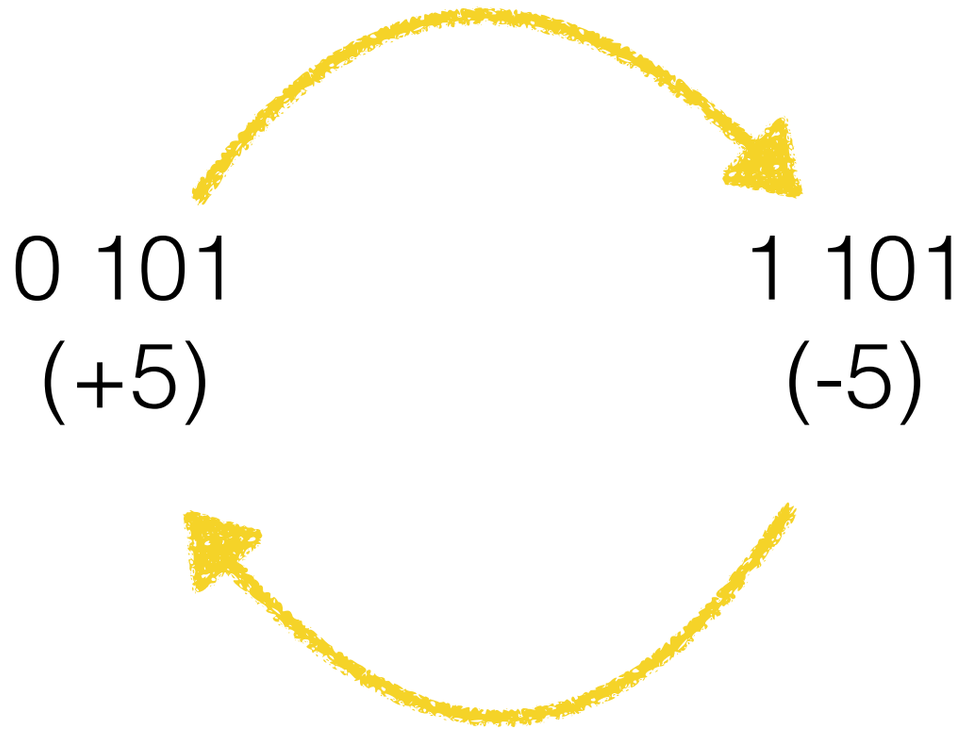
Binary	Hex	Unsigned Decimal
0 000	0	0
0 001	1	1
0 010	2	2
0 011	3	3
0 100	4	4
0 101	5	5
0 110	6	6
0 111	7	7
1 000	8	8
1 001	9	9
1 010	A	10
1 011	B	11
1 100	C	12
1 101	D	13
1 110	E	14
1 111	F	15

Positive Numbers

Negative Numbers

Signed Magnitude Number ***System***

Signed Magnitude Rule: To find the opposite of a positive number, just change its MSB to 1



and vice versa…

4-bit Nybble

Binary	Hex	Unsigned Decimal	**Signed Magnitude**
0 000	0	0	**+0**
0 001	1	1	**+1**
0 010	2	2	**+2**
0 011	3	3	**+3**
0 100	4	4	**+4**
0 101	5	5	**+5**
0 110	6	6	**+6**
0 111	7	7	**+7**
1 000	8	8	**-0**
1 001	9	9	**-1**
1 010	A	10	**-2**
1 011	B	11	**-3**
1 100	C	12	**-4**
1 101	D	13	**-5**
1 110	E	14	**-6**
1 111	F	15	**-7**

Does this System work With the ALU Adder?

Binary	Hex	Unsigned Decimal	**Signed Magnitud**
0 000	0	0	**+0**
0 001	1	1	**+1**
0 010	2	2	**+2**
0 011	3	3	**+3**
0 100	4	4	**+4**
0 101	5	5	**+5**
0 110	6	6	**+6**
0 111	7	7	**+7**
1 000	8	8	**-0**
1 001	9	9	**-1**
1 010	A	10	**-2**
1 011	B	11	**-3**
1 100	C	12	**-4**
1 101	D	13	**-5**
1 110	E	14	**-6**
1 111	F	15	**-7**

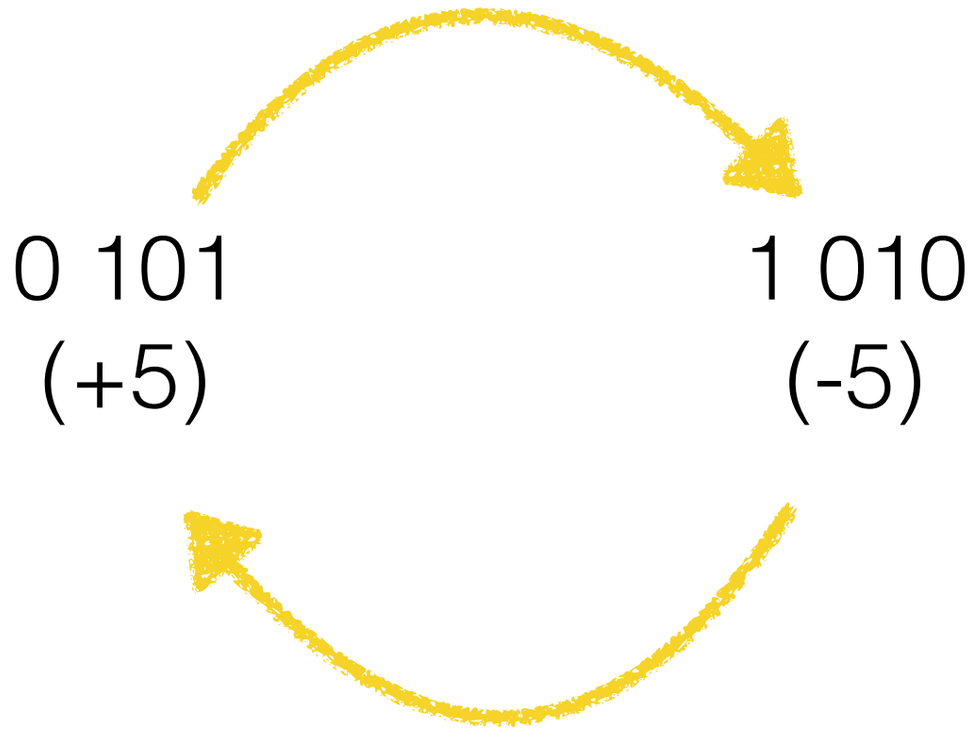
```
    3
+  -3
-----
=   0
```

```
    4
+  -1
-----
=   3
```

1's Complement

Number ***System***

1's Complement Rule: To find the opposite of a positive number, just flip all bits



and vice versa…

4-bit Nybble

Binary	Hex	Unsigned Decimal	**1's Complement**
0 000	0	0	**+0**
0 001	1	1	**+1**
0 010	2	2	**+2**
0 011	3	3	**+3**
0 100	4	4	**+4**
0 101	5	5	**+5**
0 110	6	6	**+6**
0 111	7	7	**+7**
1 000	8	8	**-7**
1 001	9	9	**-6**
1 010	A	10	**-5**
1 011	B	11	**-4**
1 100	C	12	**-3**
1 101	D	13	**-2**
1 110	E	14	**-1**
1 111	F	15	**-0**

Does this System work With the ALU Adder?

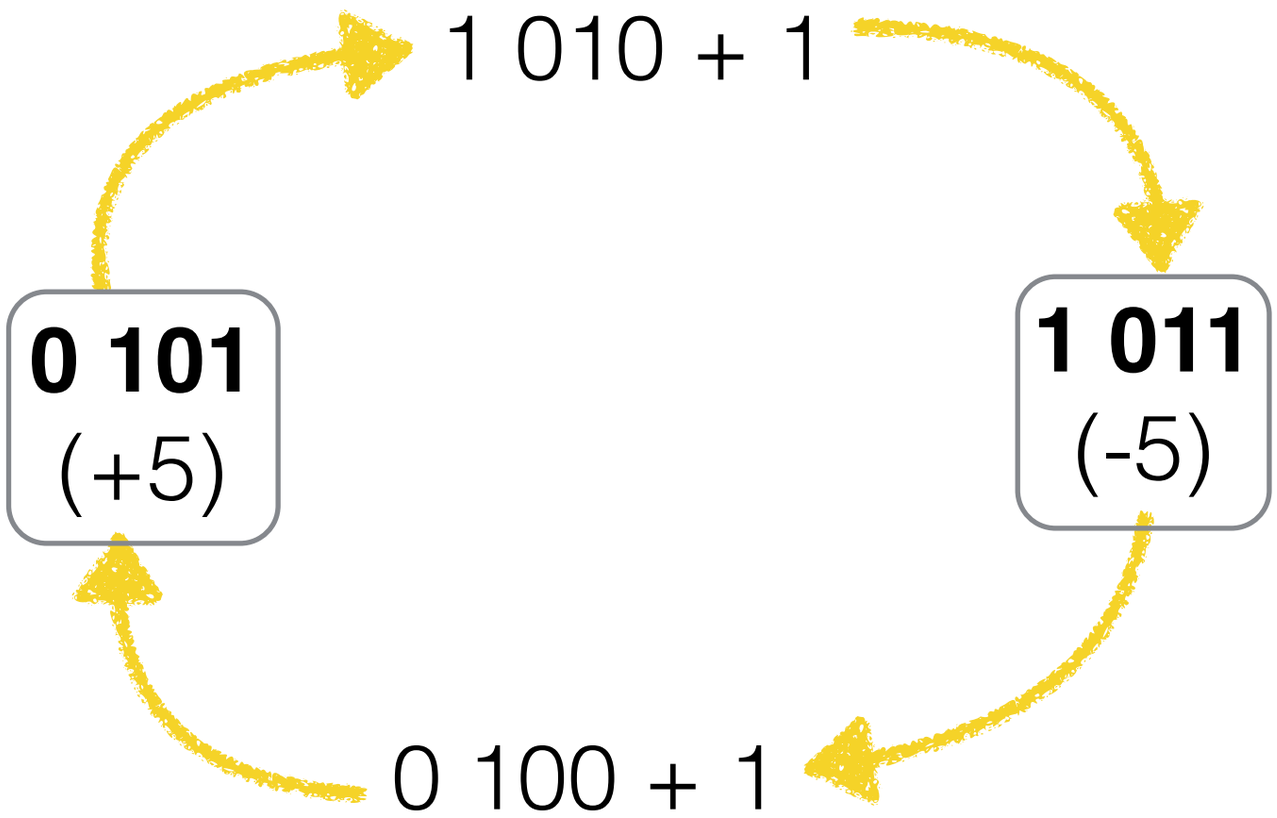
Binary	Hex	Unsigned Decimal	**1's Complement**
0 000	0	0	**+0**
0 001	1	1	**+1**
0 010	2	2	**+2**
0 011	3	3	**+3**
0 100	4	4	**+4**
0 101	5	5	**+5**
0 110	6	6	**+6**
0 111	7	7	**+7**
1 000	8	8	**-7**
1 001	9	9	**-6**
1 010	A	10	**-5**
1 011	B	11	**-4**
1 100	C	12	**-3**
1 101	D	13	**-2**
1 110	E	14	**-1**
1 111	F	15	**-0**

```
   3
+ -3
-----
   0

   4
+ -1
-----
   3

   5
+ -3
-----
   2
```

2's Complement

Number ***System***

2's Complement Rule: To find the opposite of a positive number, just flip all bits, and add 1



and vice versa…

4-bit Nybble

Binary	Hex	Unsigned Decimal	**2's Complement**
0 000	0	0	**+0**
0 001	1	1	**+1**
0 010	2	2	**+2**
0 011	3	3	**+3**
0 100	4	4	**+4**
0 101	5	5	**+5**
0 110	6	6	**+6**
0 111	7	7	**+7**
1 000	8	8	**-8**
1 001	9	9	**-7**
1 010	A	10	**-6**
1 011	B	11	**-5**
1 100	C	12	**-4**
1 101	D	13	**-3**
1 110	E	14	**-2**
1 111	F	15	**-1**

Does this System work With the ALU Adder?

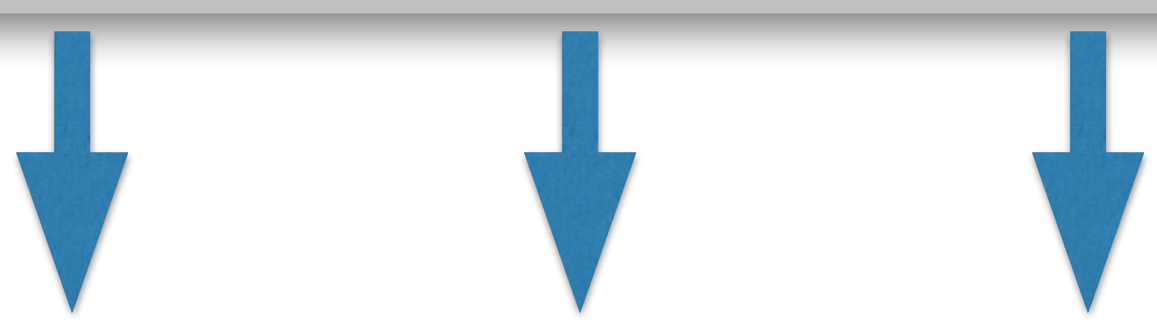
Binary	Hex	Unsigned Decimal	**2's Complement**
0 000	0	0	**+0**
0 001	1	1	**+1**
0 010	2	2	**+2**
0 011	3	3	**+3**
0 100	4	4	**+4**
0 101	5	5	**+5**
0 110	6	6	**+6**
0 111	7	7	**+7**
1 000	8	8	**-8**
1 001	9	9	**-7**
1 010	A	10	**-6**
1 011	B	11	**-5**
1 100	C	12	**-4**
1 101	D	13	**-3**
1 110	E	14	**-3**
1 111	F	15	**-1**

```
   3
+ -3
-----
=  0

   4
+ -1
-----
=  3

   5
+ -3
-----
=  2
```

Interesting Property

- What is the binary representation of **-1** as a byte?
- What is the binary representation of **-1** as a word?
- What is the binary representation of **-1** as a dword?

```
int x = 0x7fffffff - 5;

for ( int i=0; i<10; i++ )
   System.out.println( x++ );
```



Java **ints** are **signed!**

Exercises