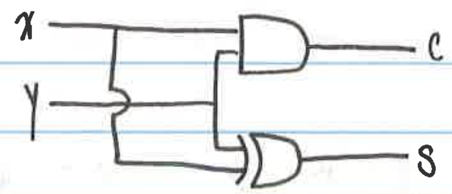
BINARY ARITHMATIC

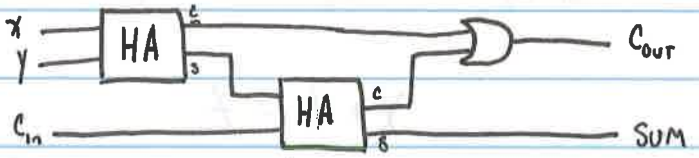
half adder



ADDS 1 BIT NUMBERS

x	+	y	=	C	S
0	+	0		0	0
0	+	1		0	1
1	+	0		0	1
1	+	1		1	0

full adder



ADDS 3 1 BIT NUMBERS

x	+	y	+	C_{in}	C_{OUT}	SUM	DEC
0	+	0	+	0	0	0	0
0	+	0	+	1	0	1	1
0	+	1	+	0	0	1	1
0	+	1	+	1	1	0	2
1	+	0	+	0	0	1	1
1	+	0	+	1	1	0	2
1	+	1	+	0	1	0	2
1	+	1	+	1	1	1	3

adding more bits

JUST LIKE WITH HALF ADDERS, COMBINING FULL ADDERS CAN ALLOW YOU TO ADD MORE BITS

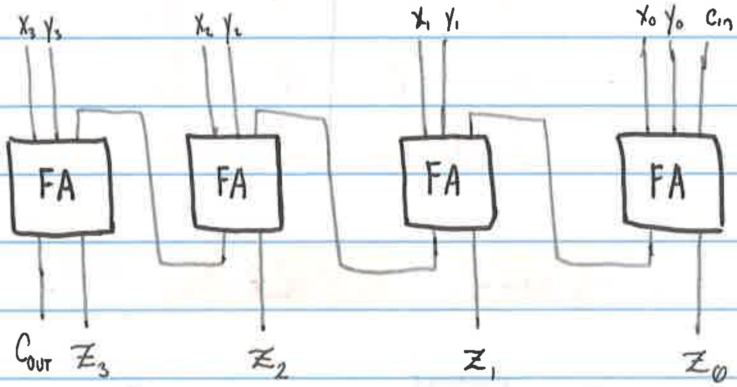
THIS ADDER ADDS 2 4 BIT NUMBERS, $X + Y$

$X =$

1	1	Ø	Ø
x_3	x_2	x_1	x_0
2^3	2^2	2^1	2^0

$Y =$

1	1	Ø	Ø
y_3	y_2	y_1	y_0
2^3	2^2	2^1	2^0



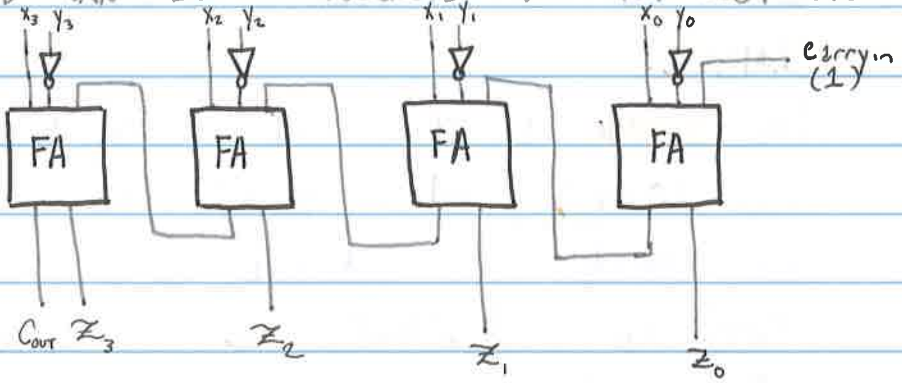
$$X + Y = Z$$

$$x_3 x_2 x_1 x_0 + y_3 y_2 y_1 y_0 = z_3 z_2 z_1 z_0$$

SUBTRACTION

- WE'RE DOING 2'S COMPOSITE WHEN WE SUBTRACT, WE'RE ACTUALLY ADDING THE NEGATIVE . . .

- WHAT WE CAN DO IS NEGATE Y WITH NOT OPERATORS



- SINCE ITS 2S COMPOSITE WE ADD 1. THATS THE CARRY IN

- SO IN ESSENCE WHAT WE ARE DOING IS...

$$X + {}^{-}y + 1 = Z$$

$$x_3 x_2 x_1 x_0 + -y_3 y_2 y_1 y_0 + 1 = Z_3 Z_2 Z_1 Z_0$$

ADDER / SUBTRACTOR
ALU

ARITHMATIC LOGIC UNIT

IN THE ALU, IN PLACE OF A CARRY WE HAVE A CONTROL.

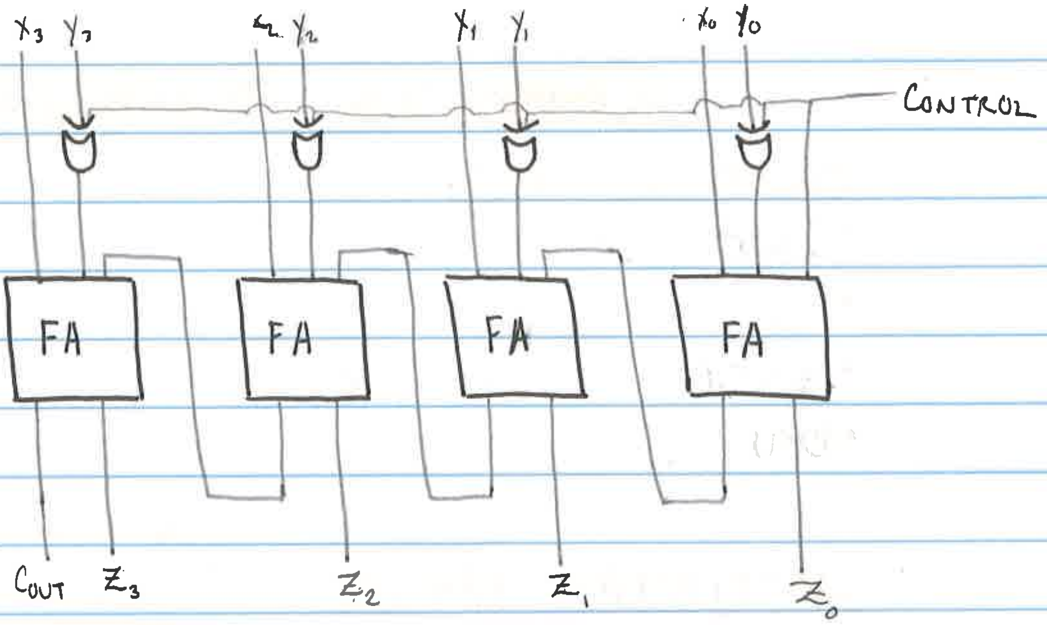
THE CONTROL CONTROLS WHETHER WE ARE ADDING OR SUBTRACTING. WHEN WE ADD, the CONTROL = 0. WHEN WE SUBTRACT THE CONTROL = 1.

USING XOR STATEMENTS WE CAN USE THE CONTROL TO NEGATE Y.

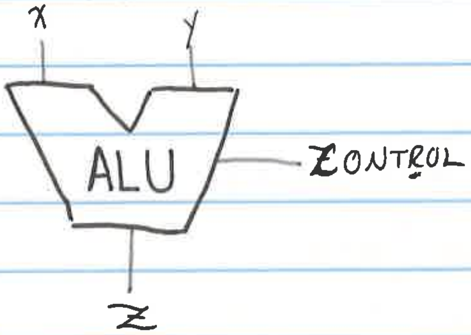
	CONTROL	Y	CONTROL ⊕ Y
ADDING	0	0	0
	0	1	1
SUBTRACTING	1	0	1
	1	1	0

SO WHEN CONTROL = 1, $\neg y$ = CONTROL ⊕ Y

SINCE THE CONTROL IS ALSO THE CARRY, WHEN THE CONTROL = 1, WHICH IS ALSO SUBTRACTING, IT TAKES CARE OF THE 2s COMPLEMENT RULE OF +1



THIS CAN ALSO BE EXPRESSED AS . . .



THE ALU IS USED IN THE CPU

Computer Organization

MAJOR ELEMENTS:

- INPUT
- OUTPUT
- MEMORY
- CPU

INPUT / OUTPUT DEVICES:

KEEP IN MIND, IN A COMPUTER, THE KEYBOARD, SCREEN + MOUSE ARE NOT NECESSARY

STORAGE DEVICES

- DISKS
- TAPES
- USB MEMORY DEVICES

COMMUNICATION DEVICES

- NETWORK INTERFACE
- TERMINAL
- KEYBOARD, SCREEN, MOUSE, ETC

MEMORY:

- REGULAR ARRAY OF BYTES (OR WORDS)
- EACH HAS A NUMERIC ADDRESS
- LOCATIONS 0 THROUGH 2^n-1 for n address bits
- GROUPS OF BYTES OFTEN TRANSFERED TOGETHER

* A NOTE ON VARIABLES AND MEMORY *

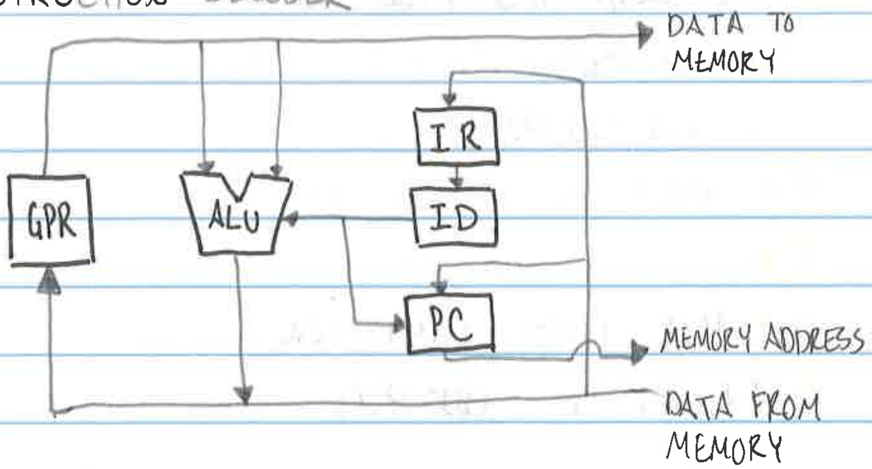
- VARIABLES ARE AN ALIAS FOR A POINT IN MEMORY
- VARIABLES ARE NOT WHAT IS STORED IN THE MEMORY POINT
- FOR EXAMPLE: x IS NOT 5 ; x IS where 5 IS STORED

CPU

CENTRAL PROCESSING UNIT

MAJOR COMPONENTS

- ALU: ARITHMETIC-LOGIC UNIT
- REGISTERS
  - PC: PROGRAM COUNTER OR INSTRUCTION POINTER
  - IR: INSTRUCTION REGISTER
  - GPR: GENERAL PURPOSE REGISTER OR ACCUMULATOR
- ID: INSTRUCTION DECODER



THE PROCESS:

THE DATA FROM MEMORY IS SENT TO THE PROGRAM COUNTER AND THE INSTRUCTION REGISTER.

THE PC KEEPS EVERYTHING ON TRACK. IT KEEPS THE FLOW OF THE PROGRAM.

THE IR HAS A LIST OF INSTRUCTIONS WITHIN IT. IT ACTUALLY TELLS THE COMPUTER WHAT TO DO.

THE ID DECODES THE IR'S INSTRUCTIONS. IN THIS CASE IT DECODES THE INSTRUCTIONS INTO THE CONTROL FOR THE ALU.

THE ALU AND GPR DO THEIR THANG THAT THE INSTRUCTIONS TOLD THEM TO DO AND WRITE IT TO MEMORY. THIS IS THE OUTPUT

Punch Card Coding

BEFORE LANGUAGES THERE WAS PUNCH CARD CODING WITH PHYSICAL PUNCH CARDS TO INSERT INTO A COMPUTER FOR AN OUTPUT

1) HAND WRITE YOUR CODE
2) TRANSFER EACH LINE ONTO A SEPARATE PUNCH CARD
3) PUT INTO COMPUTER
4) WAIT FOR OUTPUT