#1) Part 1

28 ÷ 2 = 0

14 ÷ 2 = 0

7 ÷ 2 = 1

3 ÷ 2 = 1

1 ÷ 2 = 1

0

Binary

padding to 16 bit

0000000000011100

Conversion to Hex by grouping

(0000)(0000)(0001)(1100)

HEX

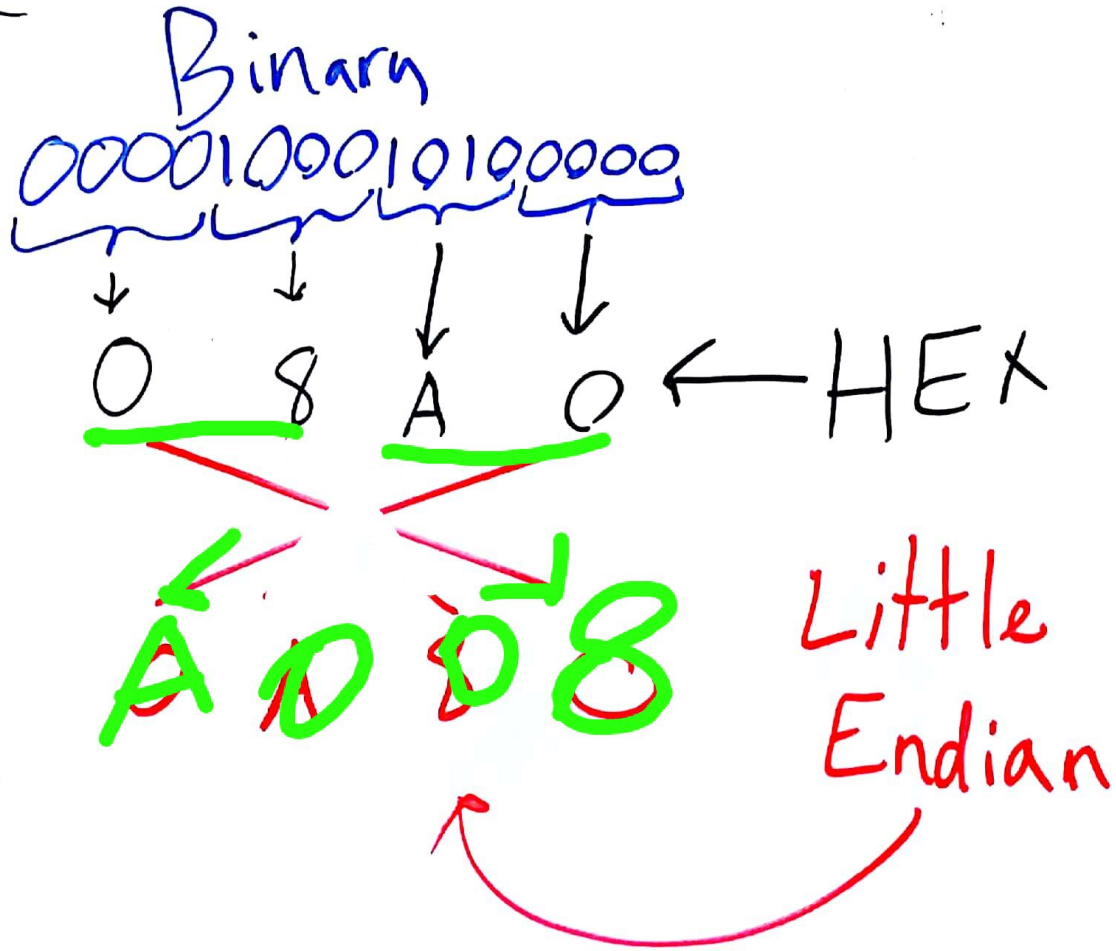
0 0 1 C

Little Endian

Lowest Address

#1) Part 2

2216 ÷ 2 = 0
1108 ÷ 2 = 0
554 ÷ 2 = 0
277 ÷ 2 = 0
138 ÷ 2 = 0
69 ÷ 2 = 1
34 ÷ 2 = 0
17 ÷ 2 = 1
8 ÷ 2 = 0
4 ÷ 2 = 0
2 ÷ 2 = 0
1 ÷ 2 = 1
0



#1) Part 3

-18675

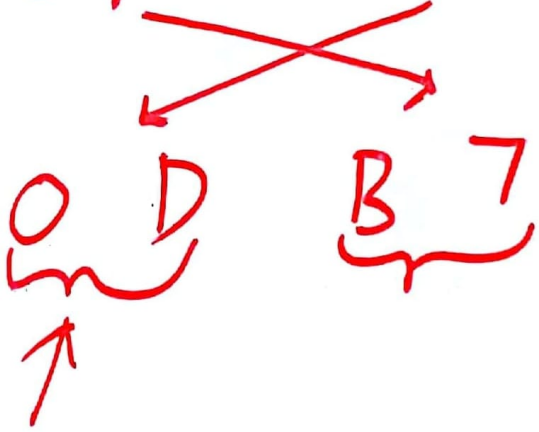
18675 ÷ 2 = 1
9337 ÷ 2 = 1
4668 ÷ 2 = 0
2334 ÷ 2 = 0
1167 ÷ 2 = 1
583 ÷ 2 = 1
291 ÷ 2 = 1
145 ÷ 2 = 1
72 ÷ 2 = 0
36 ÷ 2 = 0
18 ÷ 2 = 0
9 ÷ 2 = 1
4 ÷ 2 = 0
2 ÷ 2 = 0
1 ÷ 2 = 1

Binary

0100100011110011

1011 0111 0000 1101 ← two's complement in Binary

B 7 0 D ← two's complement in Hex



Little Endian

Least significant byte goes into smallest address

#2.)

a.) (8-6)/2

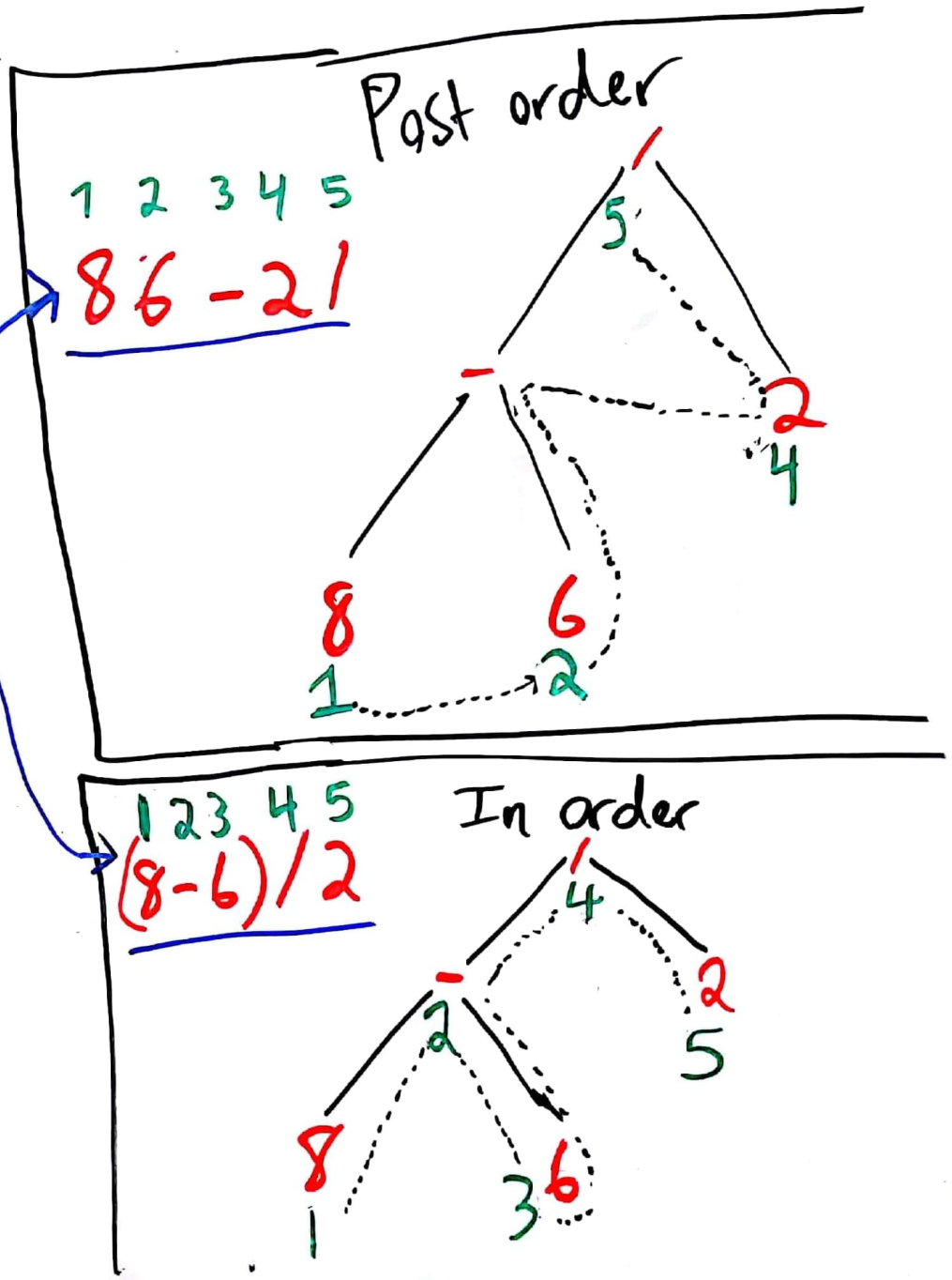
↓

(8 -6)/2

(86 -)/2

86-2/

answer



#2.) Part 2

(2+3)x8/10

↓

(2+3)*8/10

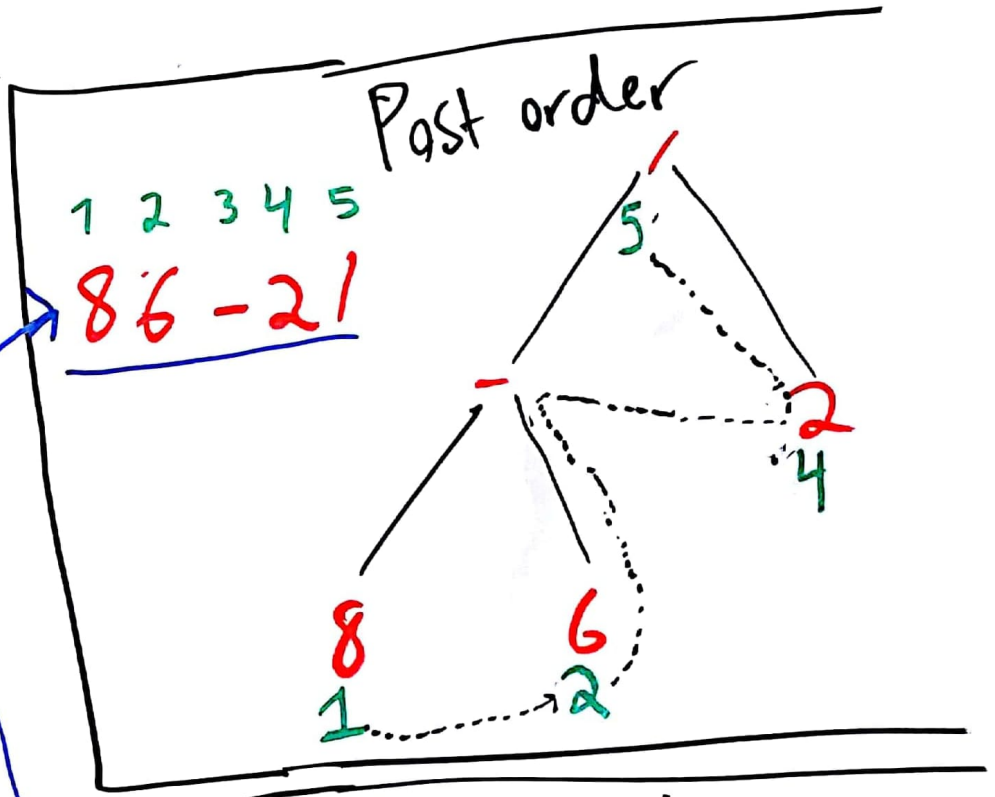
23+*8/10

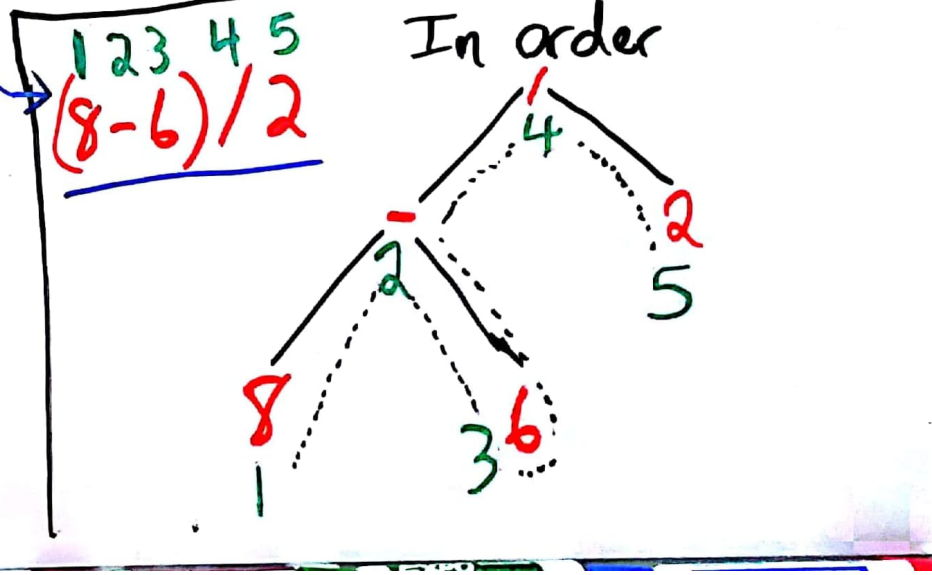
23+8*/10

23+8*10/

↑

Answer





#3) 357+21−*1++

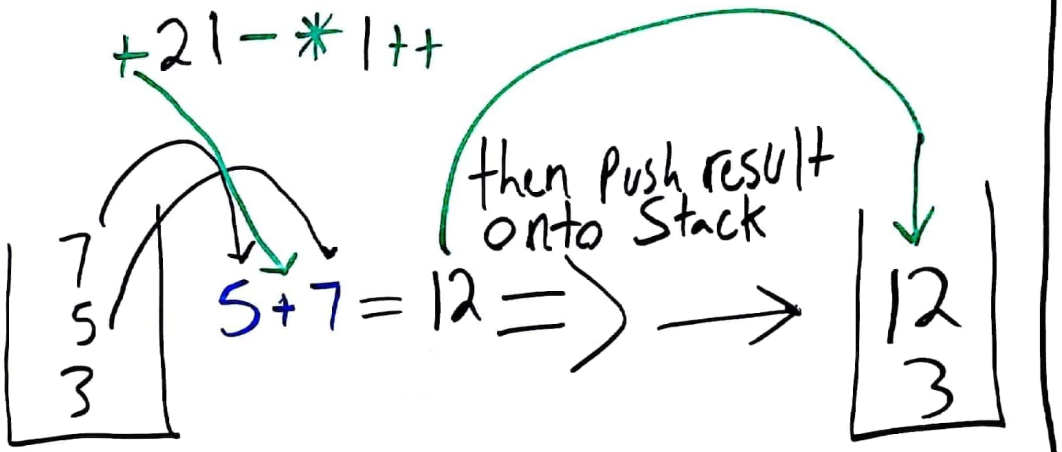
Step 1: Push numbers into Stack going left to right, stop at operation

+21−*1++

7
5
3

Step 2: Pop out top two numbers and use operation on them

+21−*1++



#21

357+21−*1++

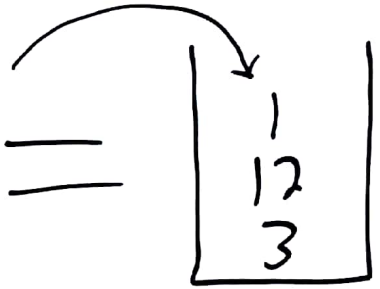
Step 3 Push numbers in until operation

21−*1++

Step 4 Pop out last two numbers then perfor qeration then Pop back in value into stack

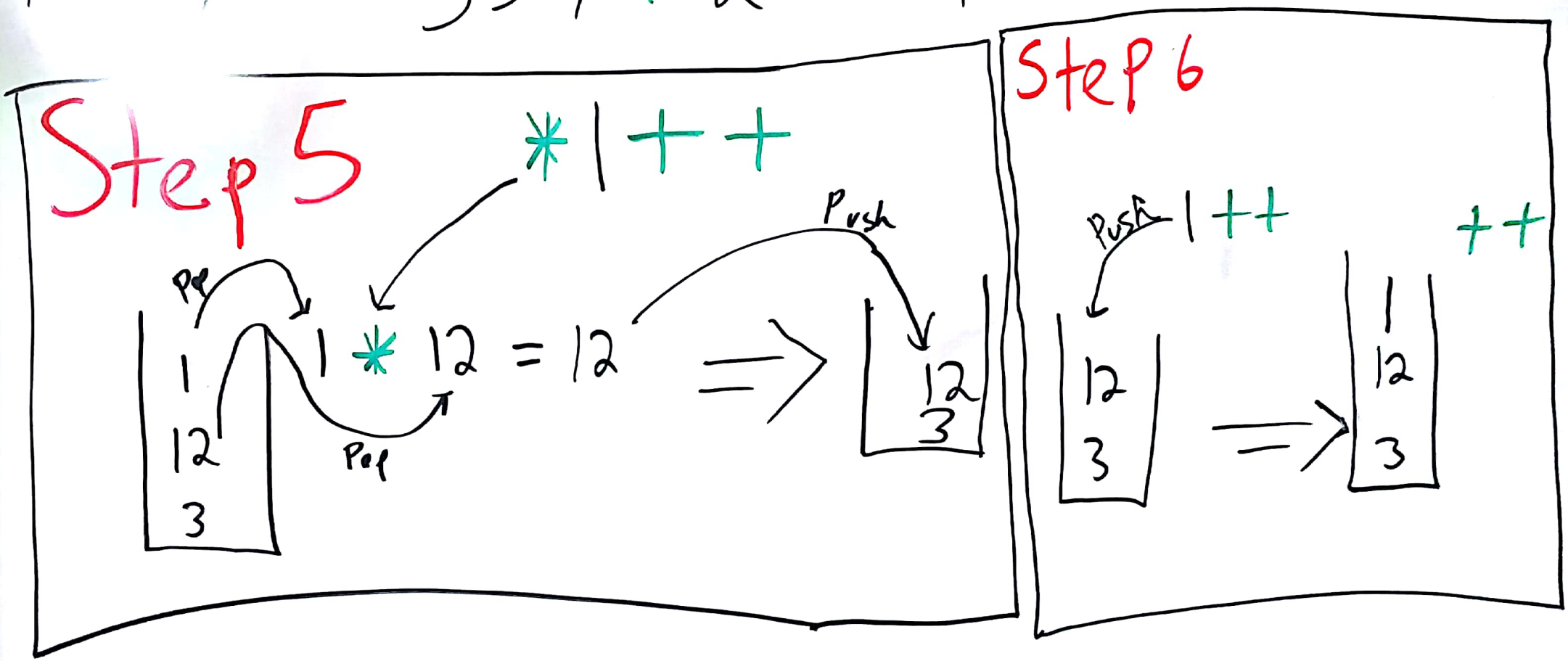
−*1++

2−1=1



#3)

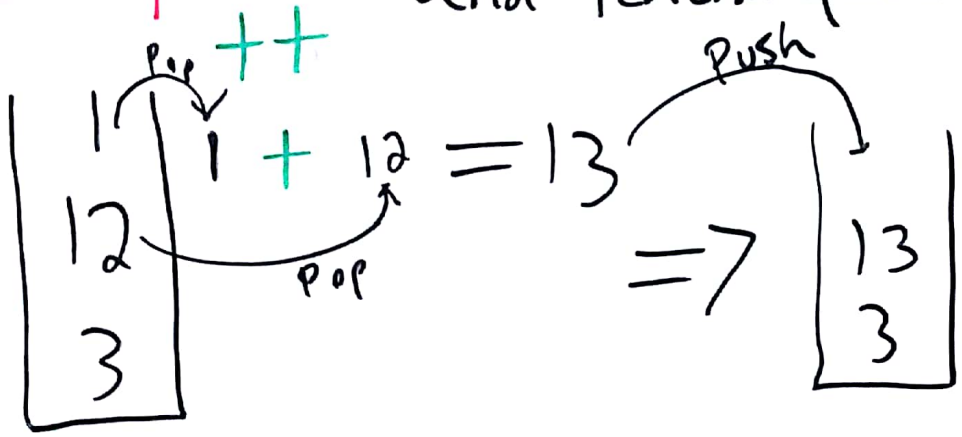
357+21-*1++



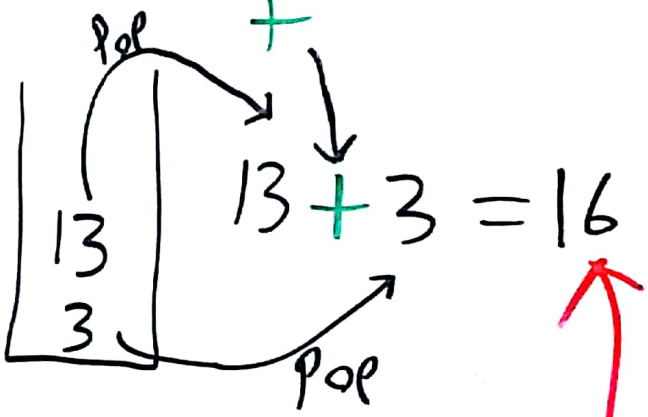
#3.) 3 5 7 + 2 1 − * 1 + +

Step 7

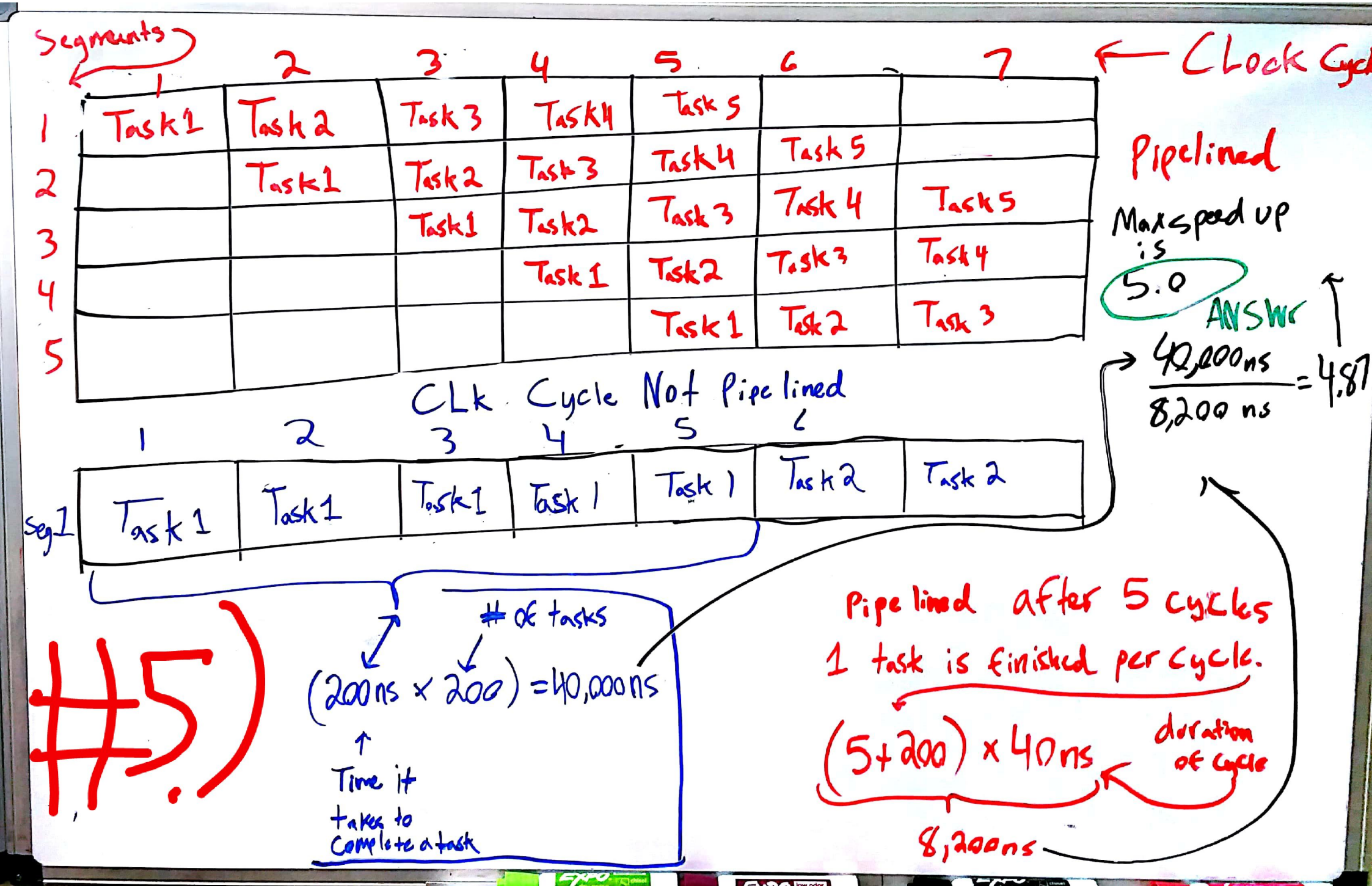
Pop two top number and Perform operation



Step 8



ANSWER


Segments
1 2 3 4 5 6 7
← Clock Cycl
1 Task 1 Task 2 Task 3 Task 4 Task 5
2 Task 1 Task 2 Task 3 Task 4 Task 5
3 Task 1 Task 2 Task 3 Task 4 Task 5
4 Task 1 Task 2 Task 3 Task 4
5 Task 1 Task 2 Task 3
Pipelined
Max speed up is
5.0
Answer
40,200 ns / 8,200 ns = 4.87
CLk Cycle Not Pipelined
1 2 3 4 5 6
Seg 1 Task 1 Task 1 Task 1 Task 1 Task 1 Task 2 Task 2
#5)
(200 ns × 200) = 40,000 ns
# of tasks
Time it takes to complete a task
Pipelined after 5 cycles
1 task is finished per cycle.
(5 + 200) × 40 ns
duration of cycle
8,200 ns

#7.) a.)

$X = R2 + Y$

$R4 = R2 + X$

cycle1	cycle2	cycle 3	cycle 4	cycle 5
Fetch Instruct	Decode	Fetch Y	Add & Store in X	
	Fetch Instruct	Decode	Fetch X	

→ Conflict

These two happen at the same time. But they fetching and storing to the same location.

b.)

$R1 = R2 + X$

$X = R3 + Y$

$Z = R1 + X$

CLK1	CLK2	CLK 3	CLK 4	CLK 5
Fetch INS	Decode	Fetch X	Add & Store in R1	
	Fetch ins	Decode	Fetch Y	Add & Store in X
		Fetch ins	Decode	Fetch X

↓

conflict

Trying to store and fetch from x at the same time.

#8.) Part 1

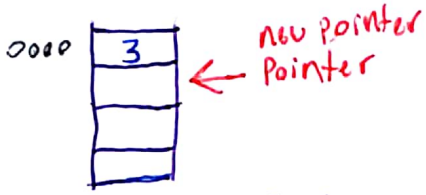
Push A

Load A Load the Value 3 into the accumulator

Store Temp store 3 into temp

Jns Push jump to the subroutine Push, then later start back at the return address

Push will store in a block of memory then n+1 on the location so the next push will be 1 above



First push at memory location n

then next push will be at n+1, Then on and on.

Push B

Load B Load the Value 5 into the accumulator

Store Temp store 5 into temp

JNS Push Jump to subroutine push

Push C

Load C Load Value 7 into accumulator

Store Temp Store 7 into temp

JNS Push Jump to the subroutine Push

#8.) Part 2

Add

JNS Add go to the Add, the Add adds the top 2 elements. Which is 5+7=12

top is bottom memory location

Push D
Load D
Store Temp
Jns Push

Push F
Load F
Store Temp
Jns

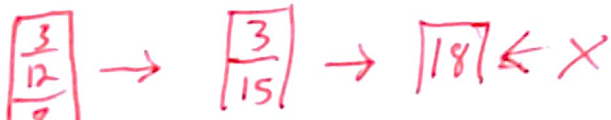
Subtract

Jns Subtract go the subtract subroutine which subtracts the values stored into the top memory locations.

Subtract
Jns Subtract

Push E
Load E
Store Temp
Jns Push

Add
Jns Add
Add
Jns Add



Pop X
Jns
Halt

#8.) b.)

What is x?

Answer X = 18