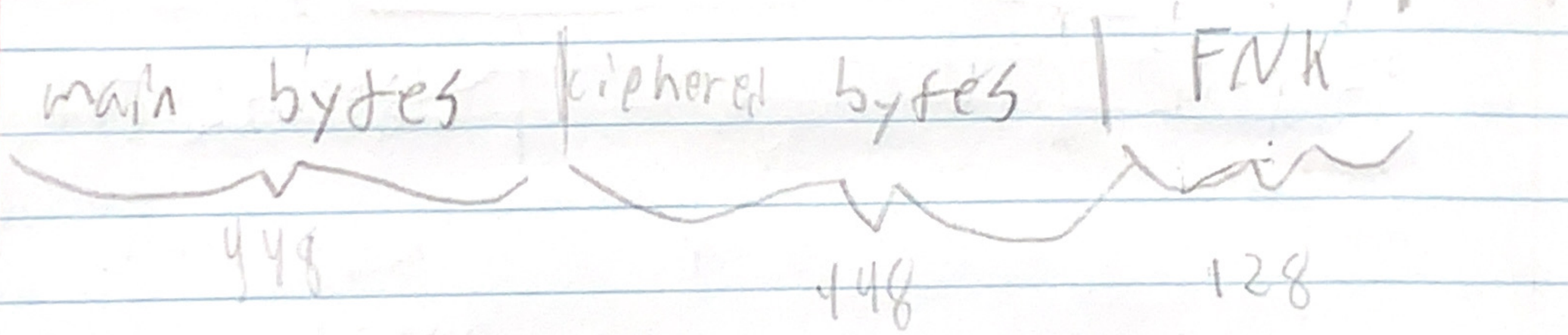
Aguing Cryptic Hash !

Hashyyy ACH-1 (work with byte[]!!!)

Speed factors!
Avoid division/sqrt
when *2 or ÷2 use bit shift to speed up
Math is speed!

block size: 1024



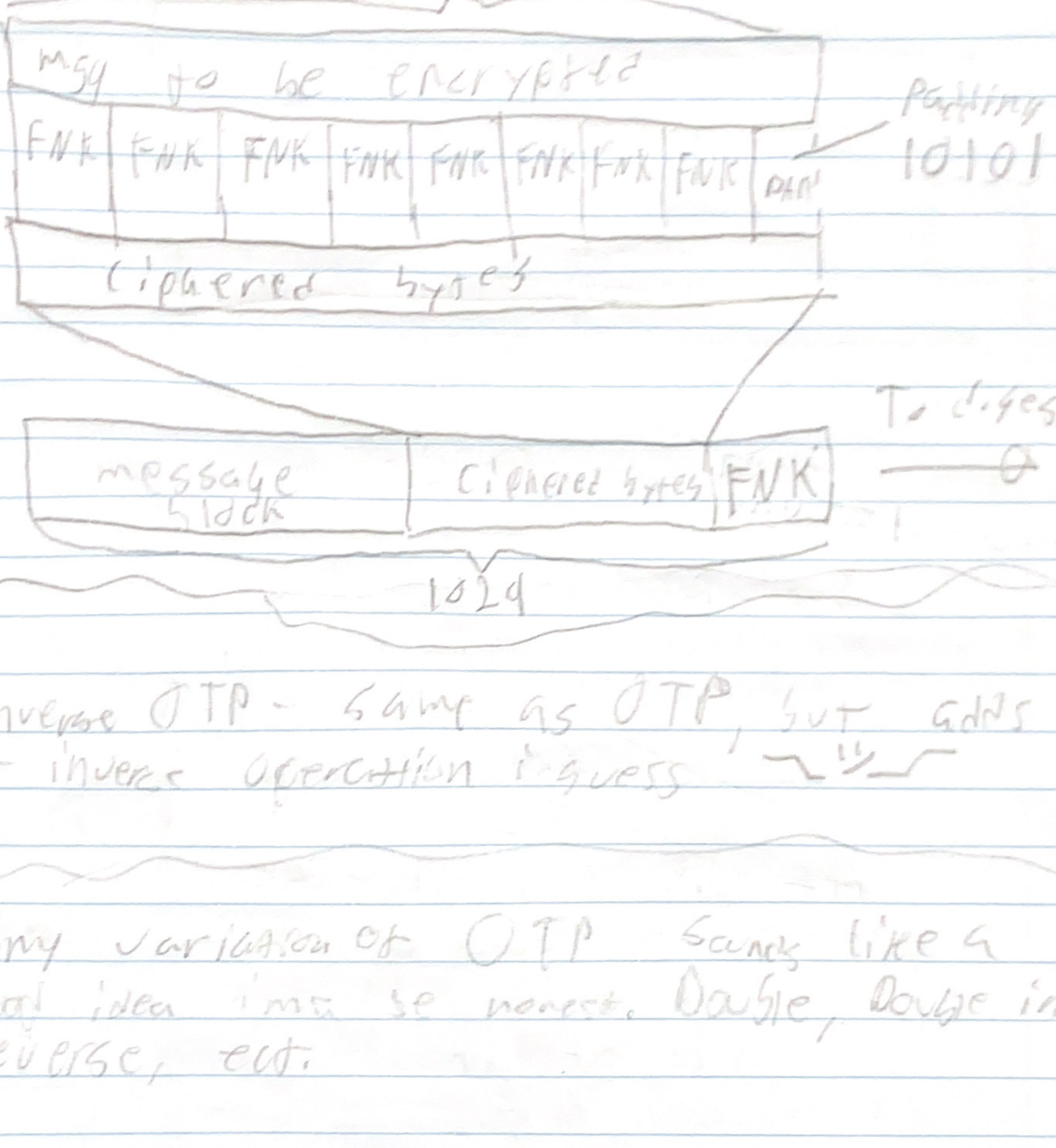
- how does it avoid collision?
- the file name gets like a seed
- the main bytes get ciphered, so replicating a similar result should technically be computationally more difficult because of added pseudo randomness
- the block size is larger

in case of no file name:
the 128 bytes get padded with 100000...

Additional notes!
- using a permutation based on the amount of 1s in a message (Boolean) name
- in case of

Possible Ciphers for 2nd Subblock

- OTP - Key is filename repeated and padded as needed. OTP is logical XOR and is therefore fast.



Note
3 FNKs and 1 pad bit in 448 bytes

- Inverse OTP - same as OTP, but adds a bit inverse operation ¯_(ツ)_/¯

- any variation of OTP sounds like a good idea it may be worse. Double, double inverse, reverse, ect.

Forming the FNK

448 128

Ciphered msg

FNK

64 bits | Nonce file name:
sampled from padding
message block structure of
subblock 10101010...

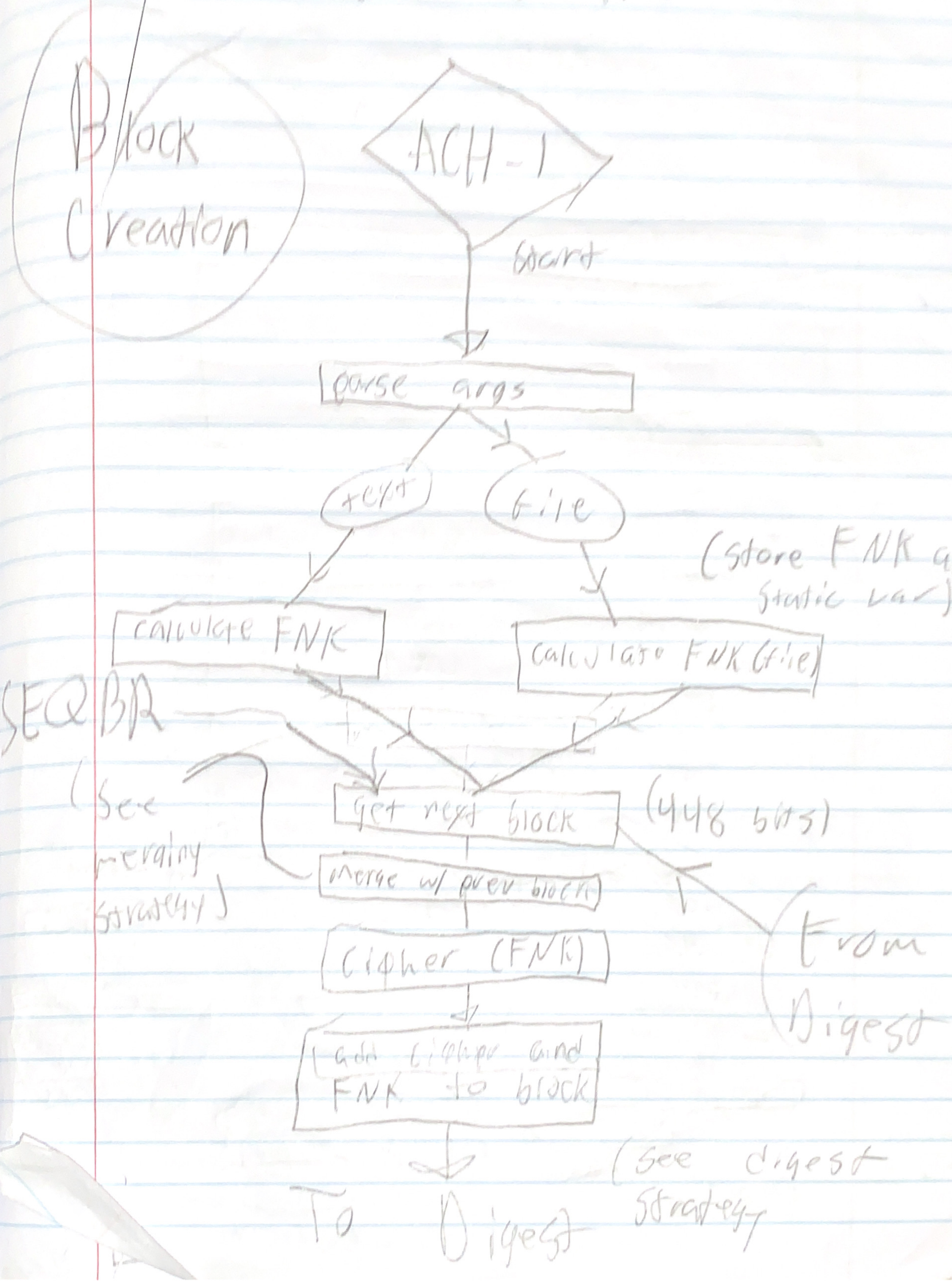
Case of file name:
Let $A_1, A_2, \ldots, A_i$ be 64 bit sections of our file name. The 64 bits of this section is represented by.

$$A_i \oplus A_{i+1}$$

loop for length n and index i

< bit manipulation bad!

Flowchard


Block creation
ACH-1
Board
parse args
text
file
(store FNK a static var)
calculate FNK
calculate FNK (file)
SEQBR
(see reading strategy)
Get next block
(448 bits)
merge w/ prev block
Cipher (FNK)
add cipher and FNK to block
From Digest
To Digest
(see digest strategy

this works

~~byte reading (limited to 32 bits)~~

forming the FNK

(revised)

148 byte

128 byte

FNK

64 byte

64 byte

64 bytes sampled from message sub block, then...

Padding Structure for no file name

0x.AA (10101010)

1) Sample 8 from 80th bit into 64 bit (add OTP)

less

greater

same

16 ... 48 ...

64 ... 60 ...

64

pad

dupe

dupe

64

64

64 pad

OTP and add

64 64

FNK Formed

OTP model

dupe

64

64

OTP add

64 64

64 - 64

FNK formed

64

64

64 pad

OTP and add

FNK Formed

$00001_{10} = 2$ <<

00100 = 4

00101 = 5

01010 ≠ 10

6 2

Careful with bit Shifts

11101 01 ☆ Im a dumbass

⊕ 11101 01

00000

Forming ciphered bytes

for each block

448

dupe

448 | 448

GFK FNK

448 | 448 | 128

dupe FNK x3

448 | 448 | 128 | 128 | 128 | 128

1408 bytes

Add FNK x3

448 | 448

384 | 128

Add

384 | 128

OTP fit

64 | 320 | 64

448

OTP fitted FNK in dupe msg

448 | [redacted]

to Digest

block created

1024

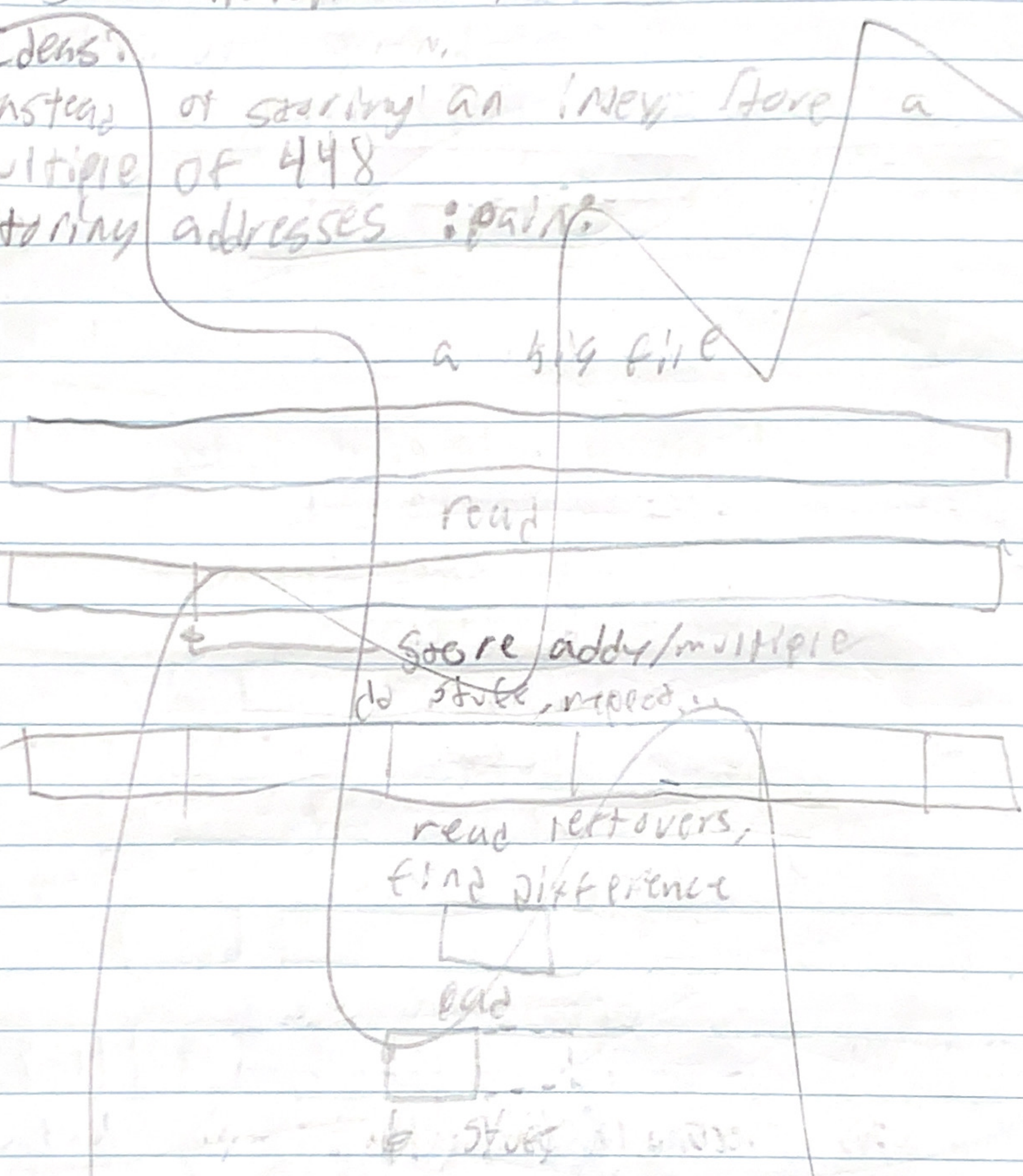
array add

448 | 448 | 128

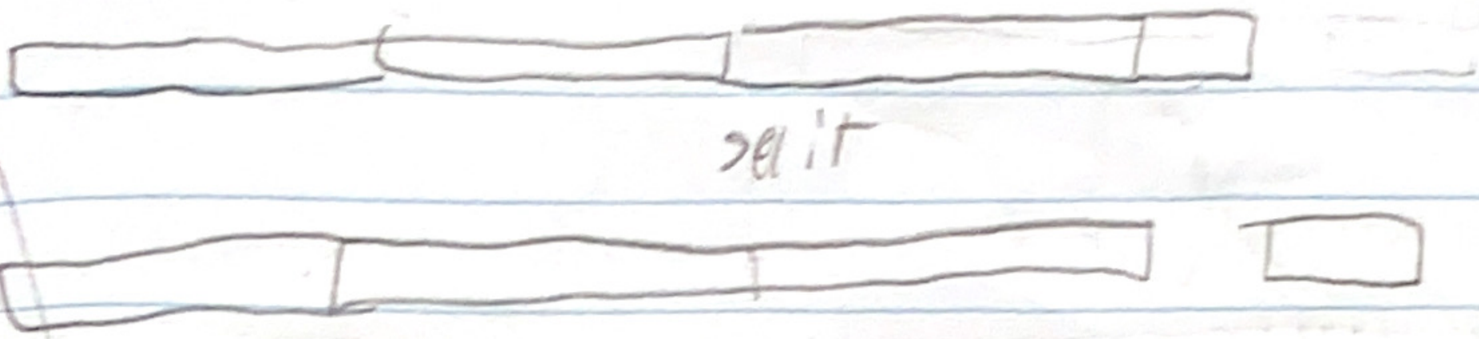
Get FNK

Sequential Byte reader

Idea:
- Instead of storing an index store a multiple of 448
- Storing addresses : pain

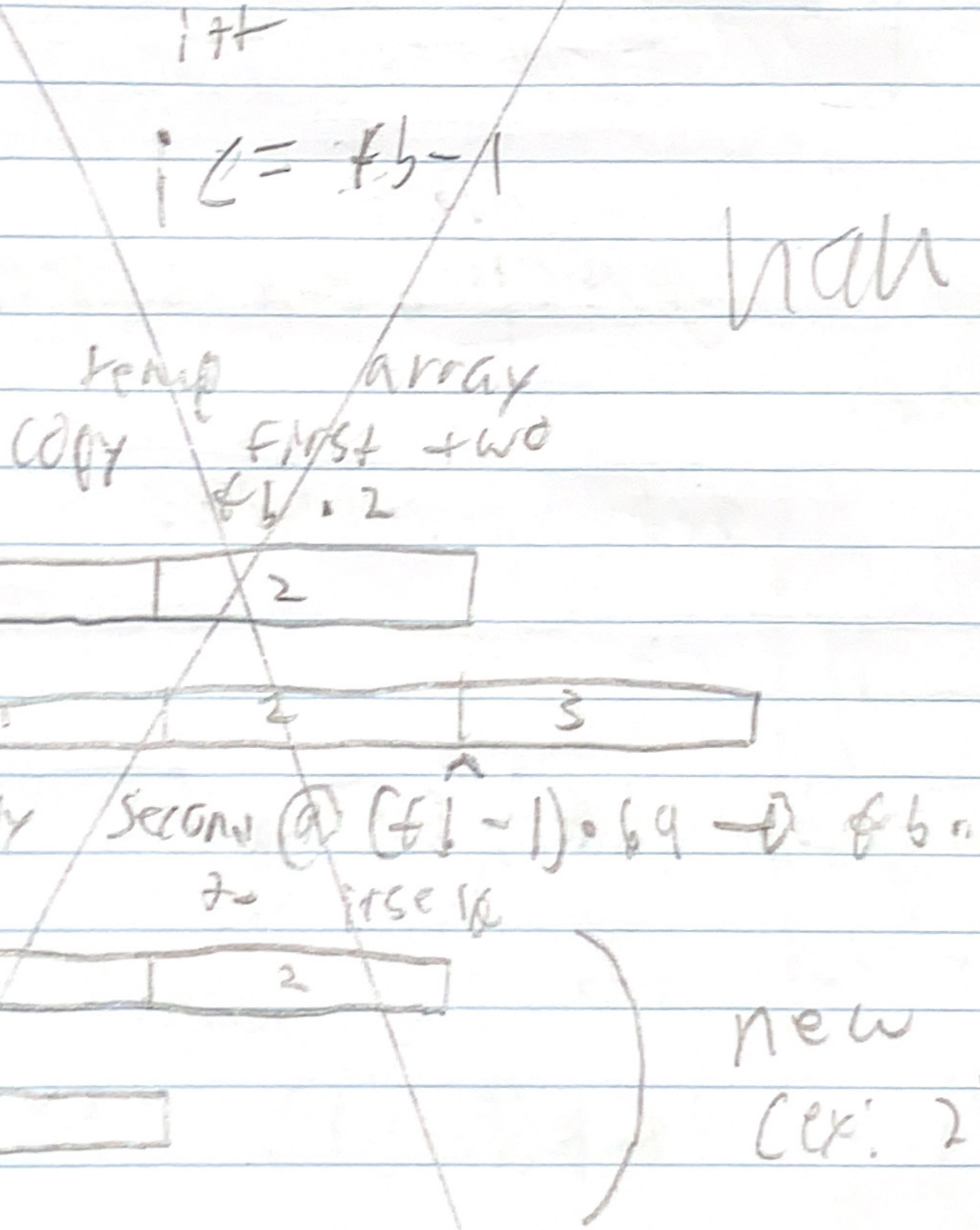


multi Schedule merge



x3

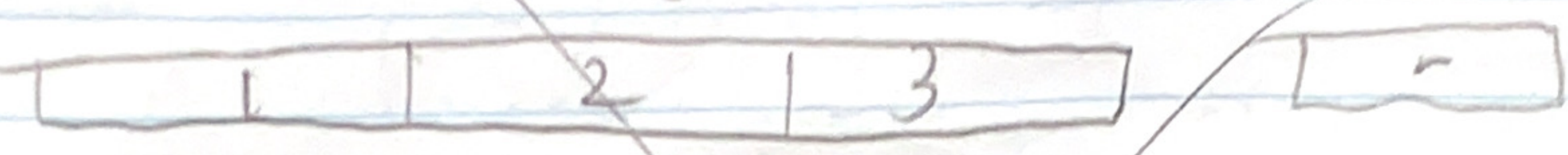
i <= fb-1



hard!

multi subblock merge

Sol'n

bnbl  rem

one merge

[byte[], byte[], byte[]]

array array!.

byte[block, address]

get fullblocks

ex. fb = 3

mb new byte[fb, 64], , ,

iterate over full blocks

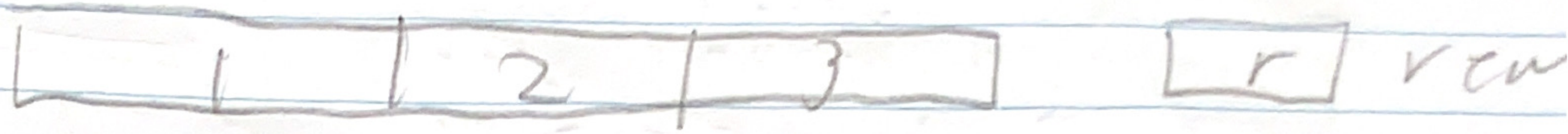
i=0 i<=(fb-1) i++

mb[i,0] = F(Array, (bnbl, (i*64), 64

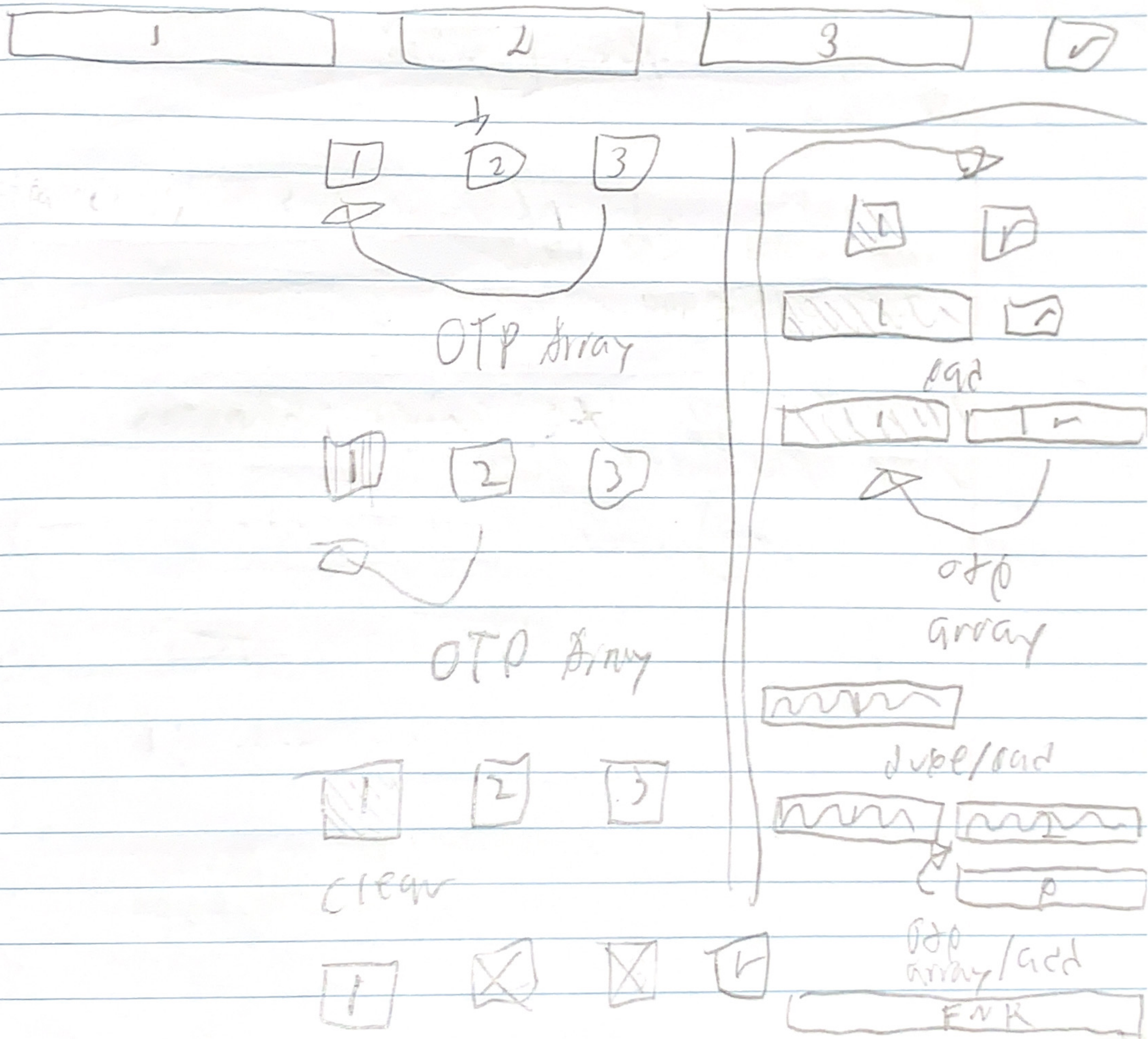
Multi Subblock merge

Split

Smgl



Jagged Array

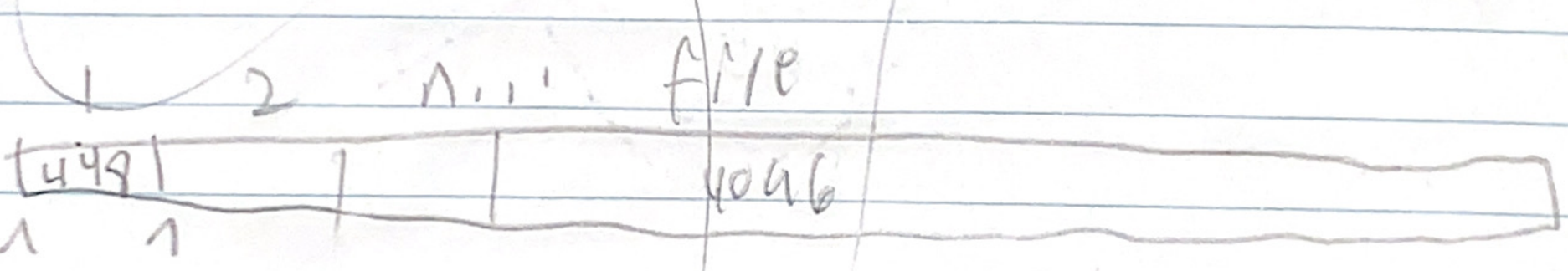


must account for at least
50 tb

Sequential byte reader

ulong address ??? ulong is enough
pointer address ??? but is
64 byte

start: 448 ; seek : 448



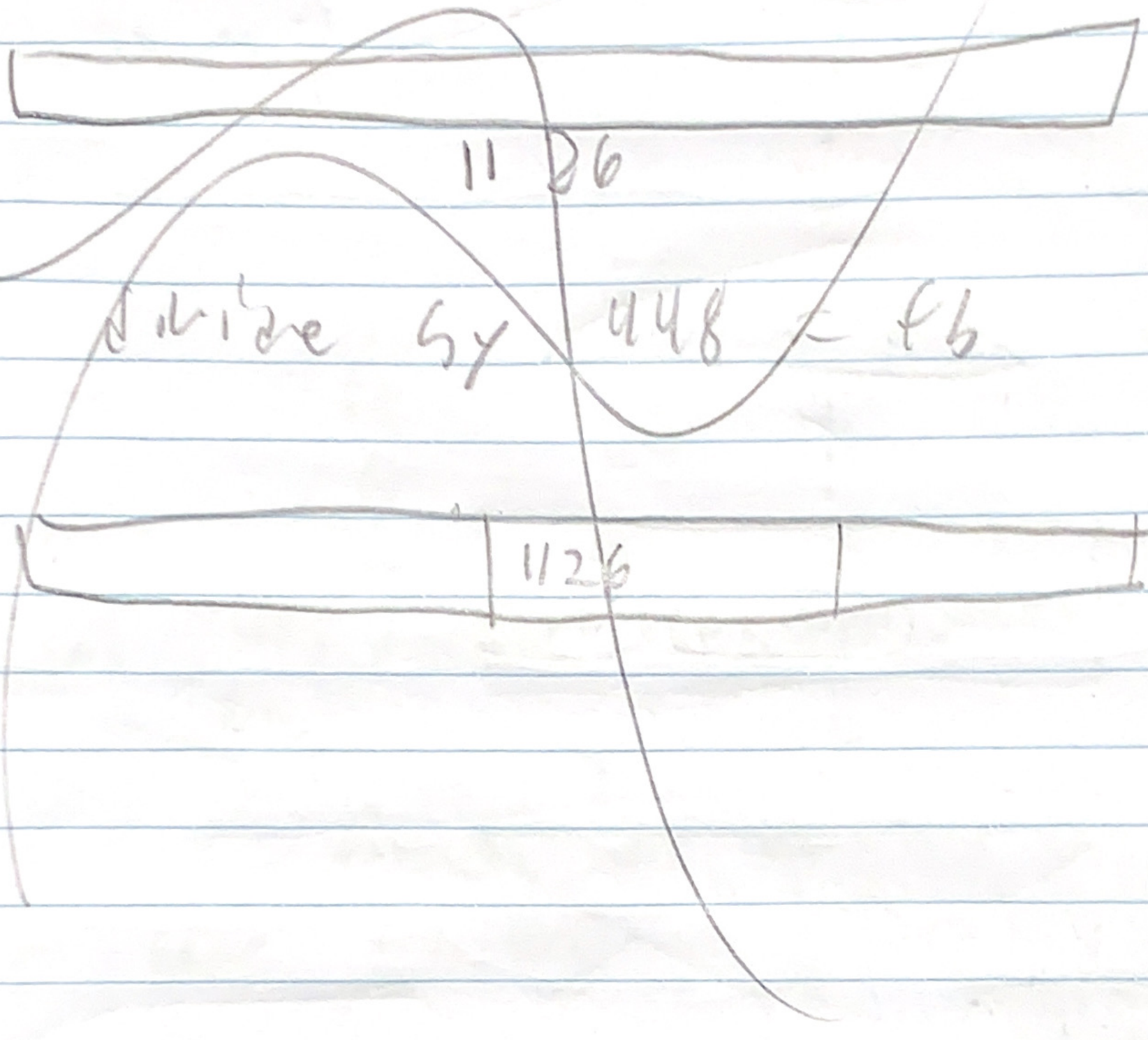
i=0 start seek
0 448

i=0 ^
start
448
i=1

loop until 1st block
== 0 // < 448

> check to see what happens when you seek past

Get length of file



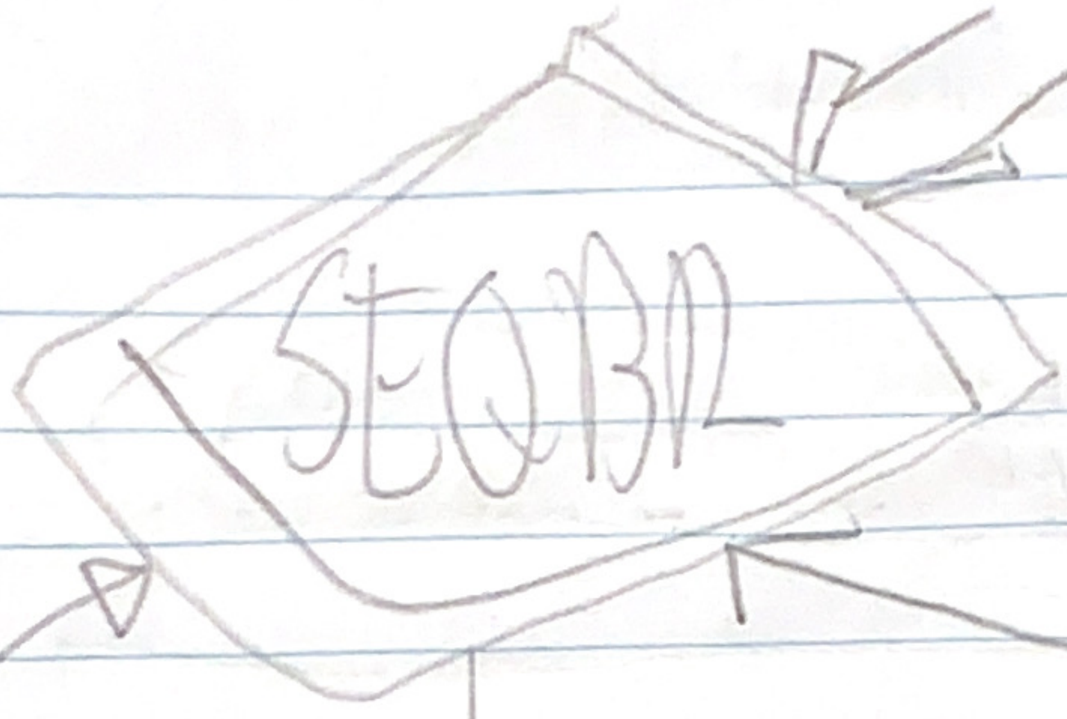
Sementical Bytereader

Generation type

(Path)

ex! Path

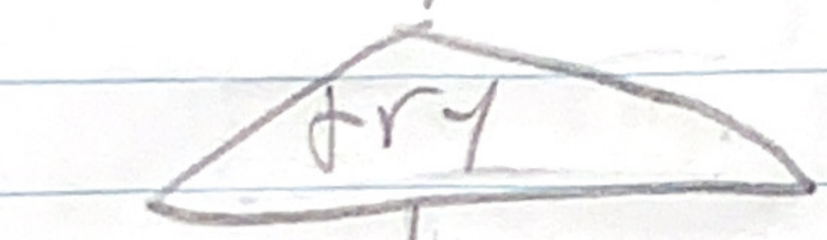
untested

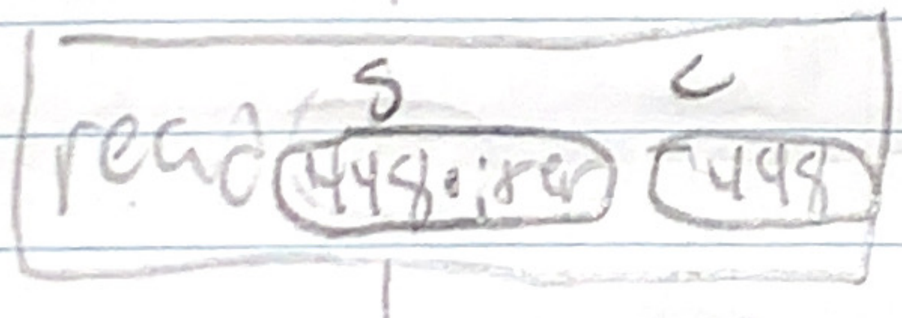


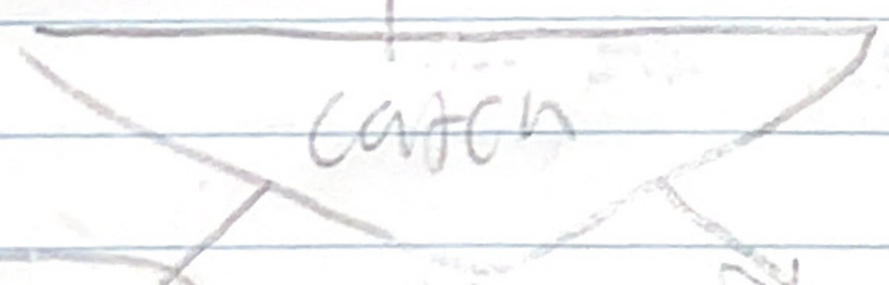
compilation

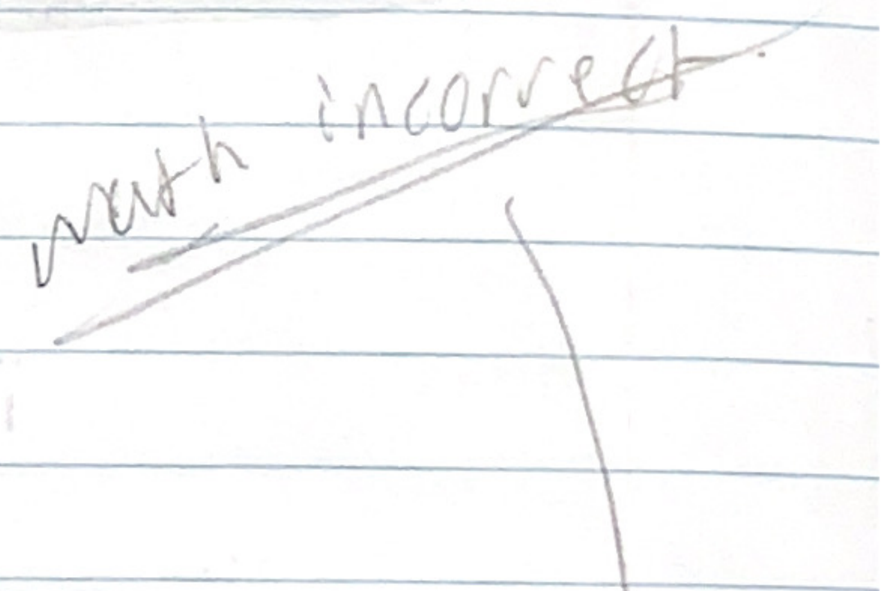
(iter)

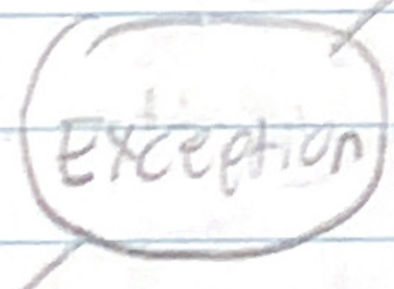
Path or data

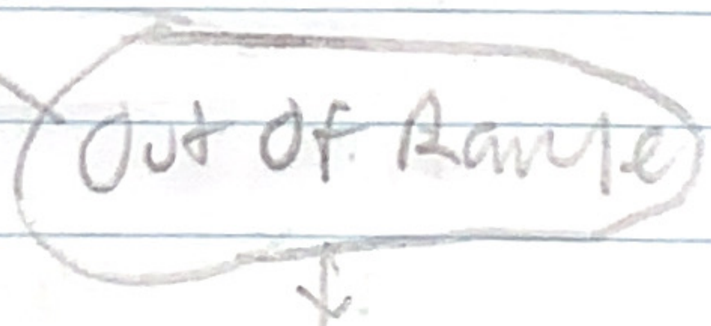


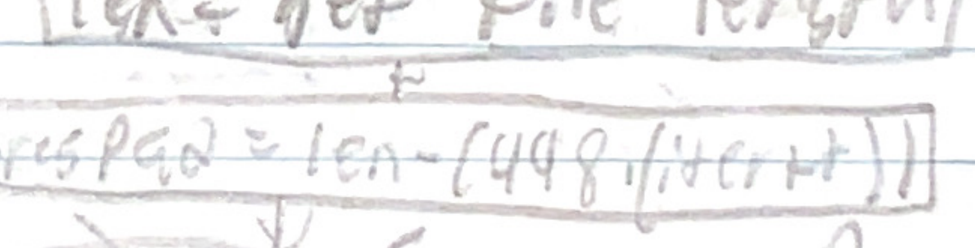


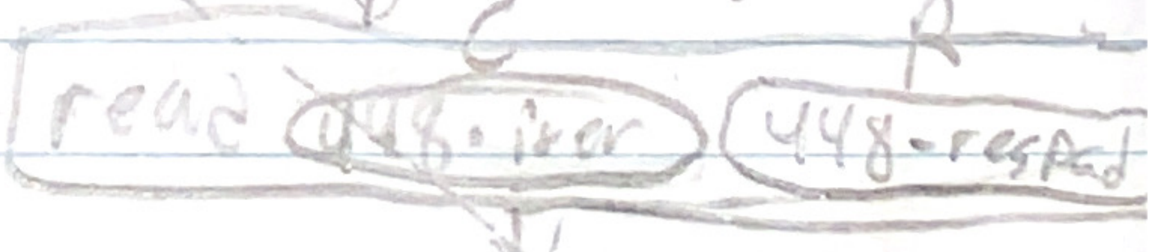




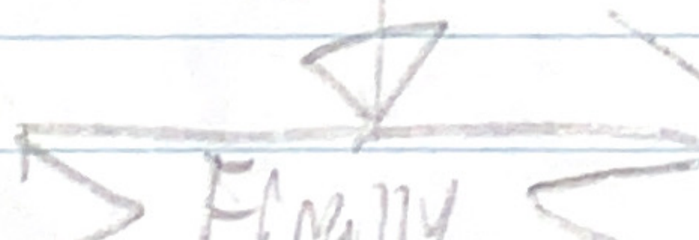


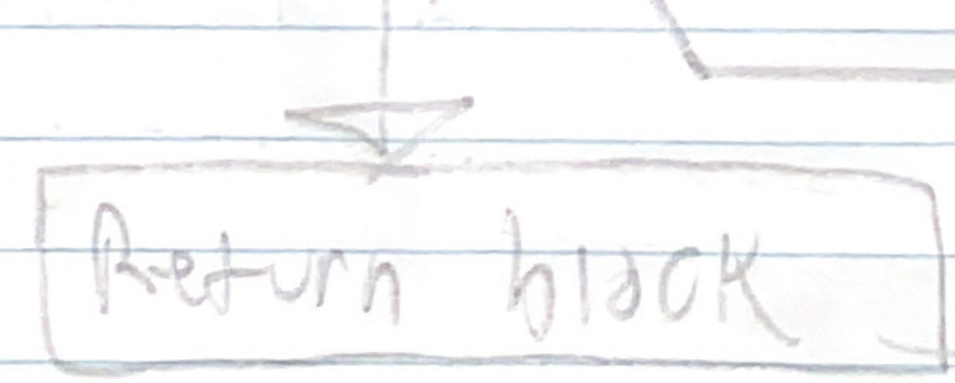






Error





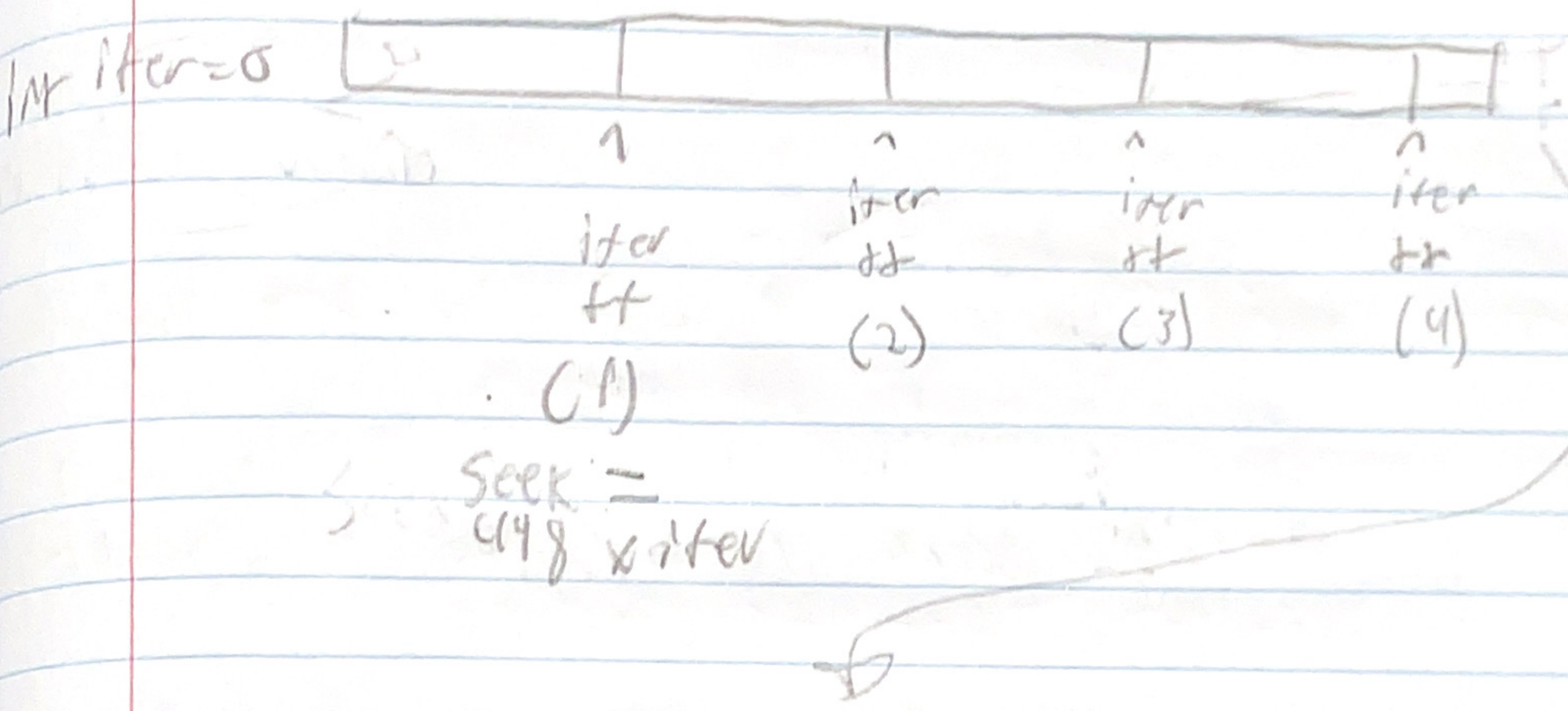
448-compression

SEQBR file block structure

unzipped

Data 2525 bytes

read 448



Seek = 448 * iter

Reading more will cause an error,

to save memory

(this way saves memory in case of lengths of multiple of 448

case error in try catch

get the total file length len

read from

len-(448*(iter+1))=respad

math incorrect

seek (iter*448)

448-respad

aka 448-(len-(448*(iter++))

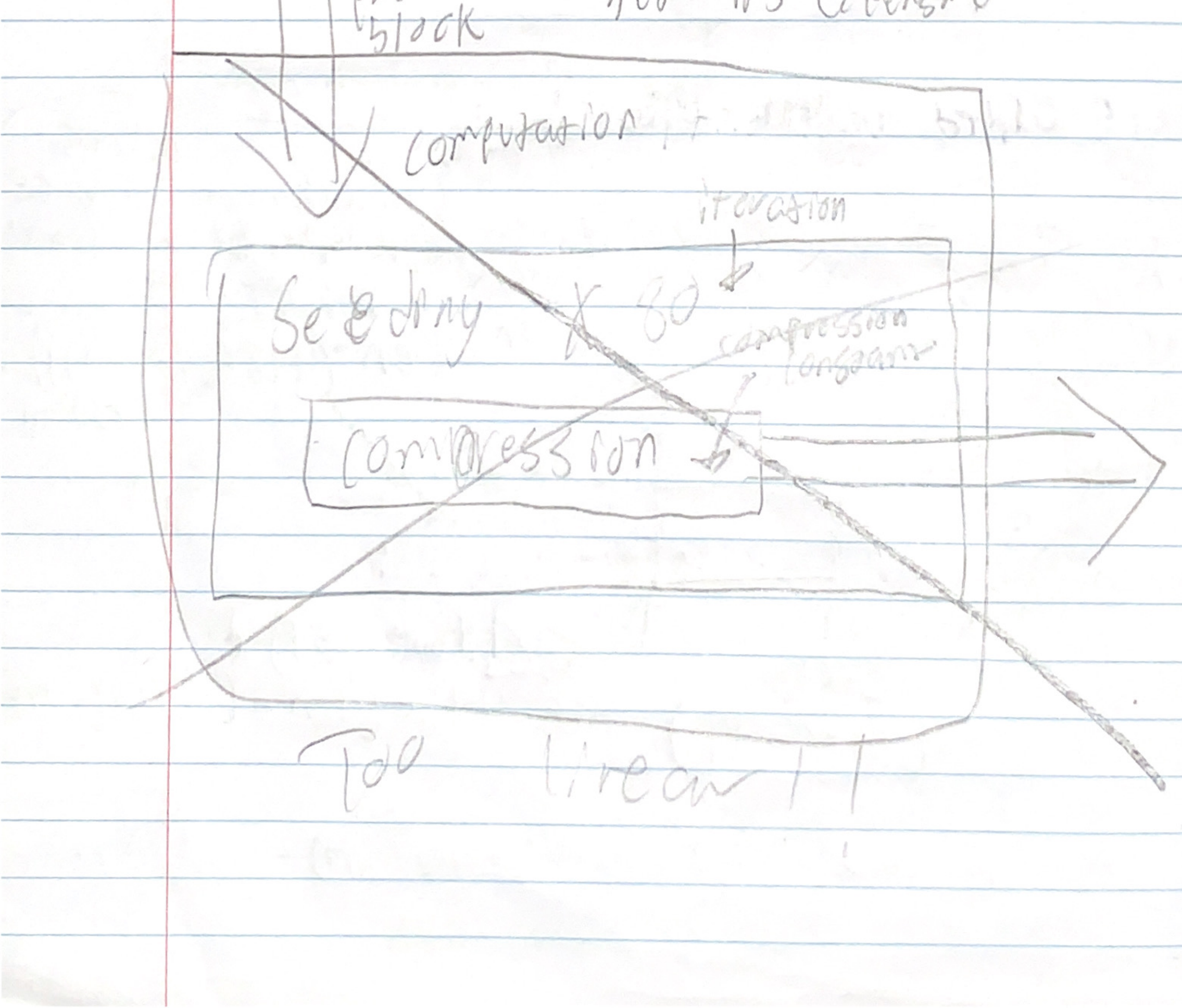
read count = ((iteration+1)*read count) - file length

Block Digestion Method

- must be pseudo random
- must be mathematically irreversible
- multiple seeding functions in series theoretically create a hashing function

Things to consider:

- byte rotation
- compression algorithm
- using math (continuous functions) to seed variables
- hashing a function to get a constants for the current computation iteration
- Addition mod 2^32 seems like a good compression method but its opposite

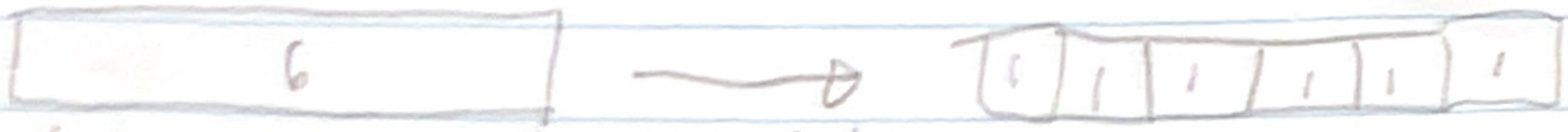


12-7-21
1000
Anders

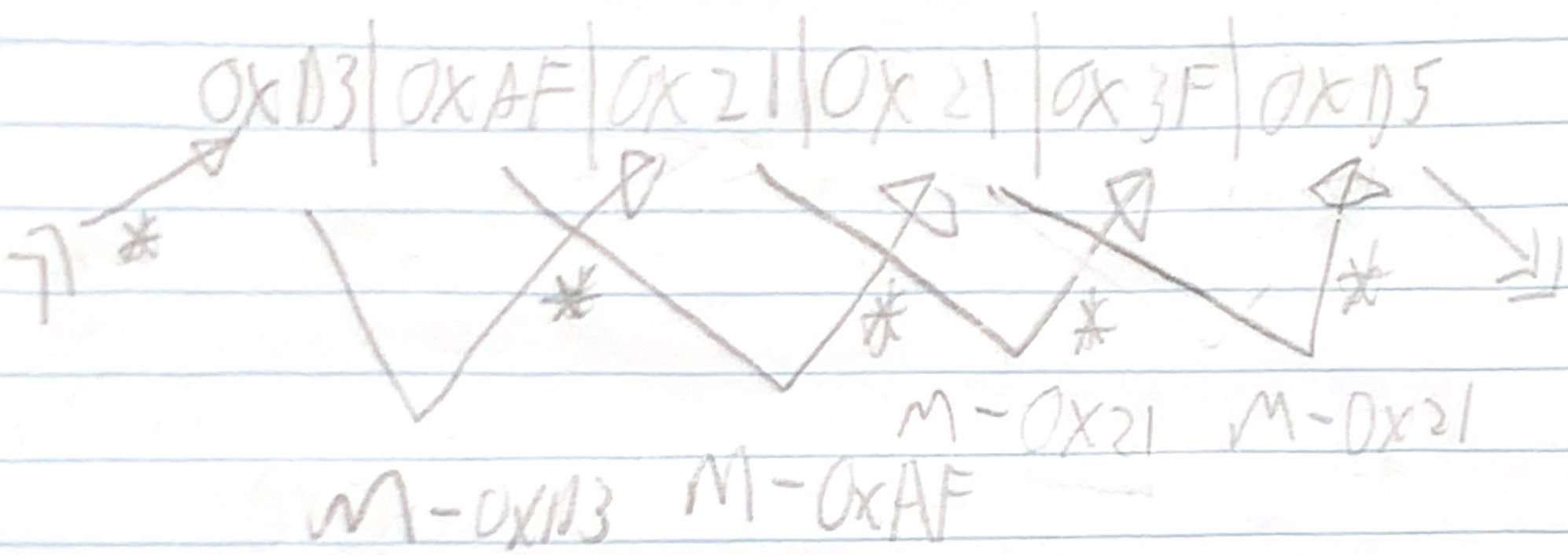
The Crazy Idea Page

these are random ideas I think might work

- Byte Jumping gets the difference



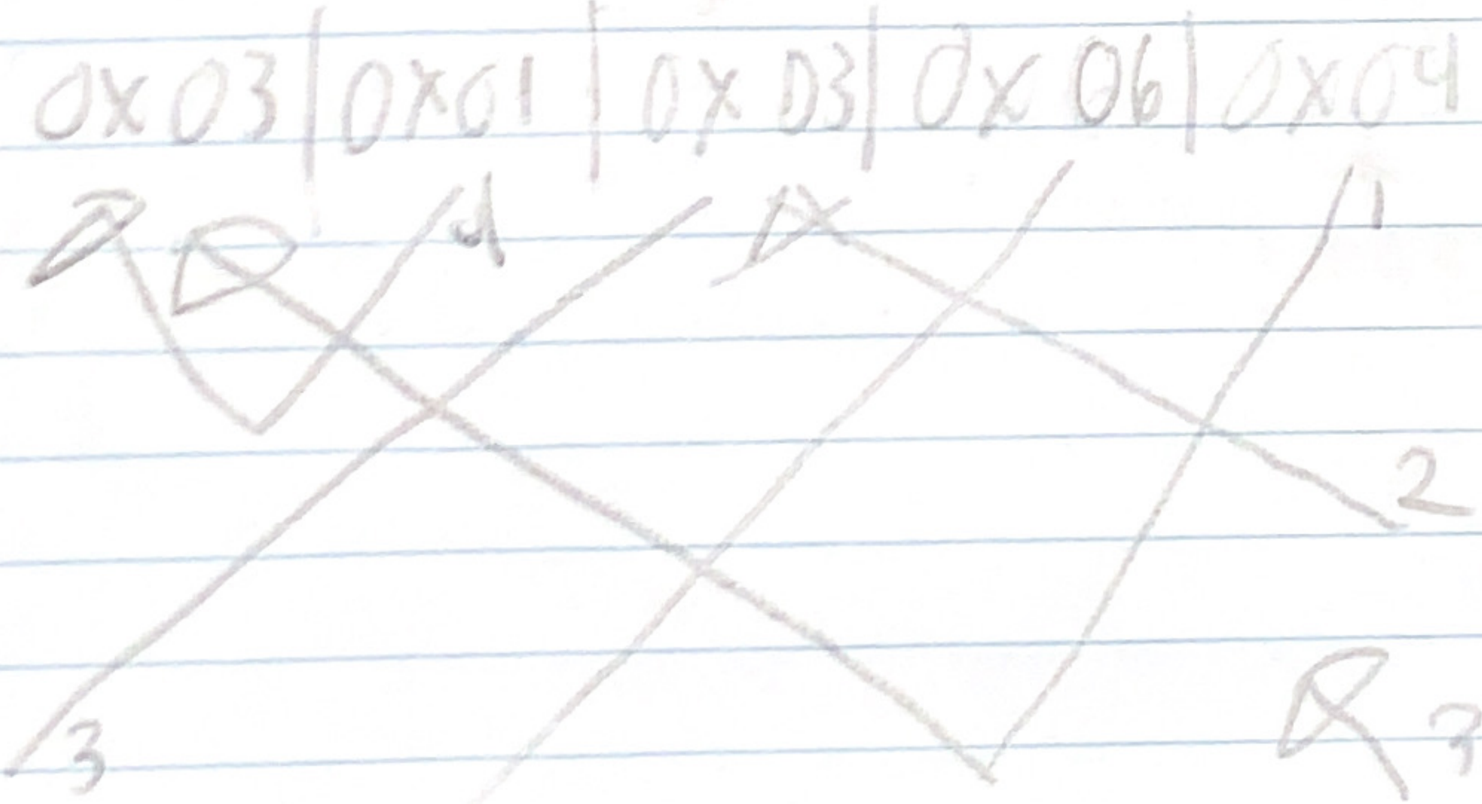
Suppose byte M = 0xFF



- byte Splicing: the byte at position n is gets OTPed with the byte at the integer representation of byte at address n, address n is 1024 - iteration.

block[n] ⊕ block[block[n] to int()]

example:

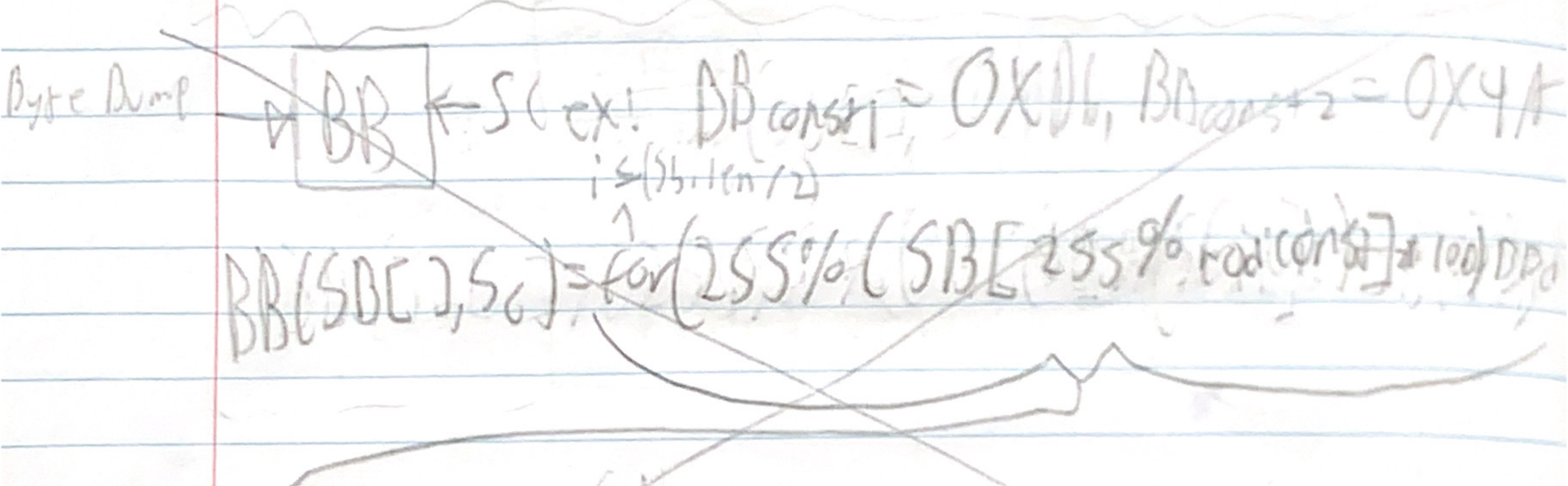


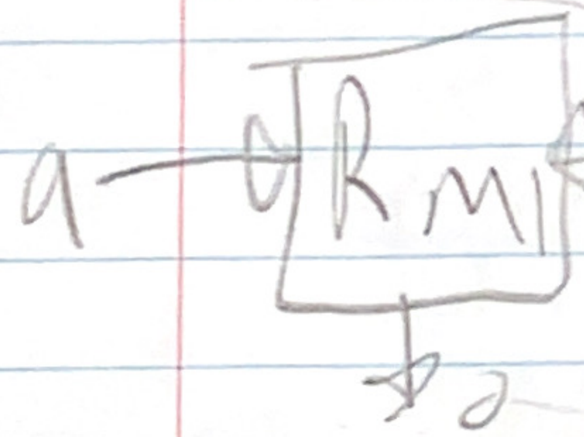
Substack digestion functions p.2

= add mod 2^8

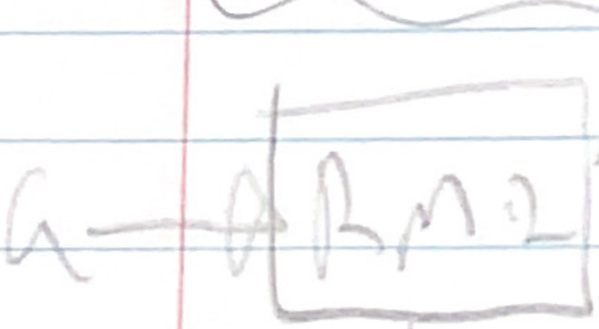
- Remarks/requirements: multi argument functions whose inputs add up to 8

Byte Dump





RM1(a, Sc) = (a<<<RMC1) ⊕ (a>>>RMC2) ⊕ (a<<<RM



ex. constants
RMC1 = 150 % Sc
RMC2 = 20 % Sc
RMC3 = 240 % Sc

Sample of constants

Crazy Ideas P. 2 | Subblock digest funs

getting a seed constant for round CompIter

GenS(in B[], CompIter) =

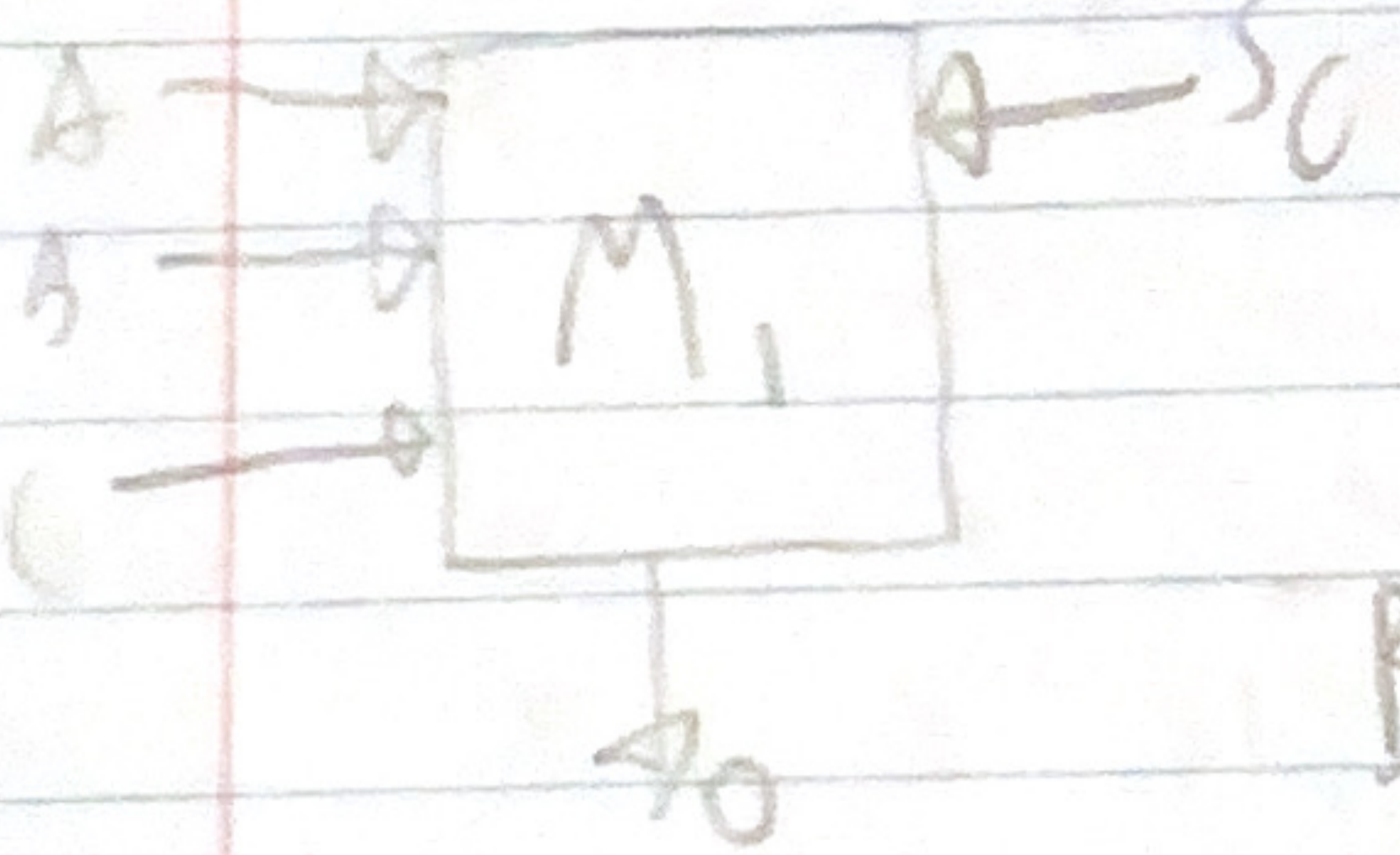
$f(i) = \sin(B[c_1] * (I/20) * \cos((I^{B[c_2]/10}) ^{1+\frac{B[c_3][?]}{100}}$

sign, power, fraction

IEEE 794 struct. 0|100000000|0000000000000000000000000

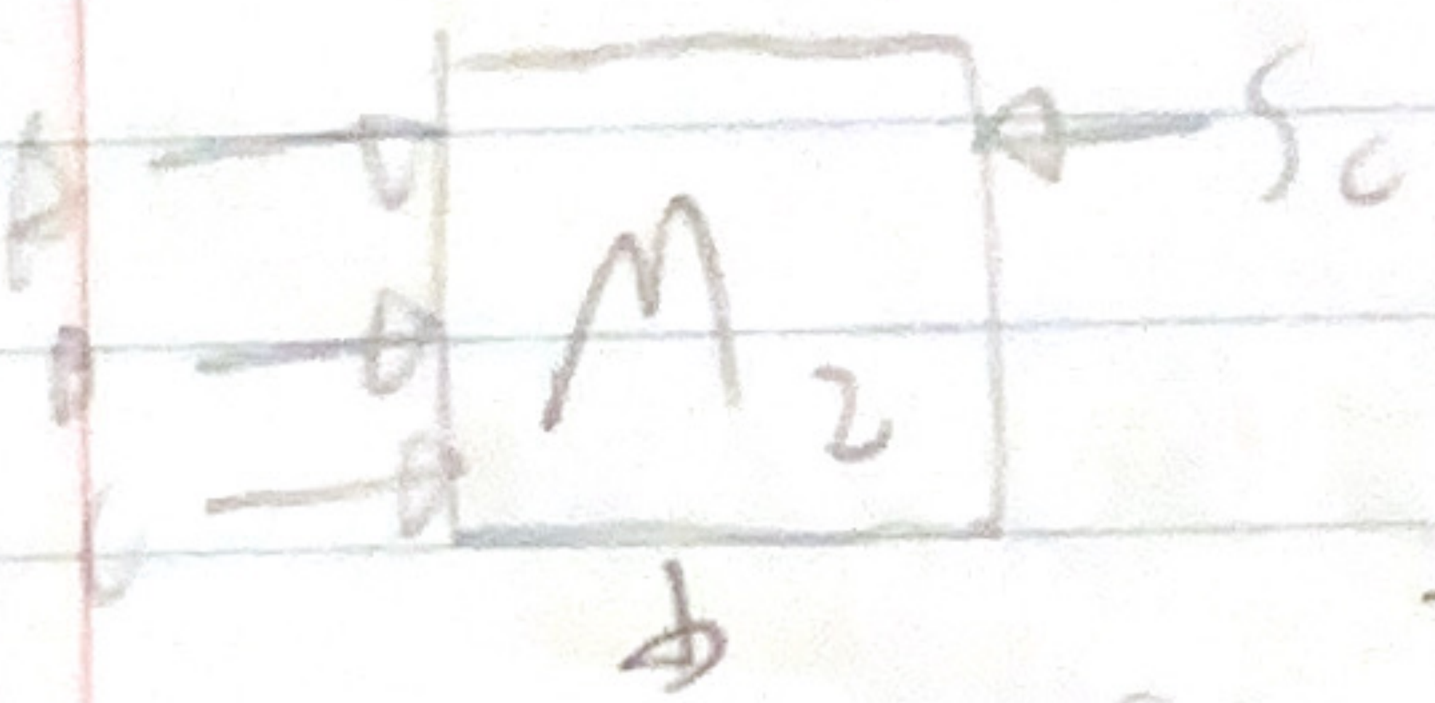
f(i) <<< 9

get first !0x0 in GetBytes(f(i))



$B_a = 255 \% (B * S_c)$

$M_1(A,B,C) = \overbrace{(A \oplus B_a)}^{T_1} \wedge \overbrace{(\neg A \oplus C)}^{T_2} \oplus \overbrace{(B \odot C)}^{T_3}$



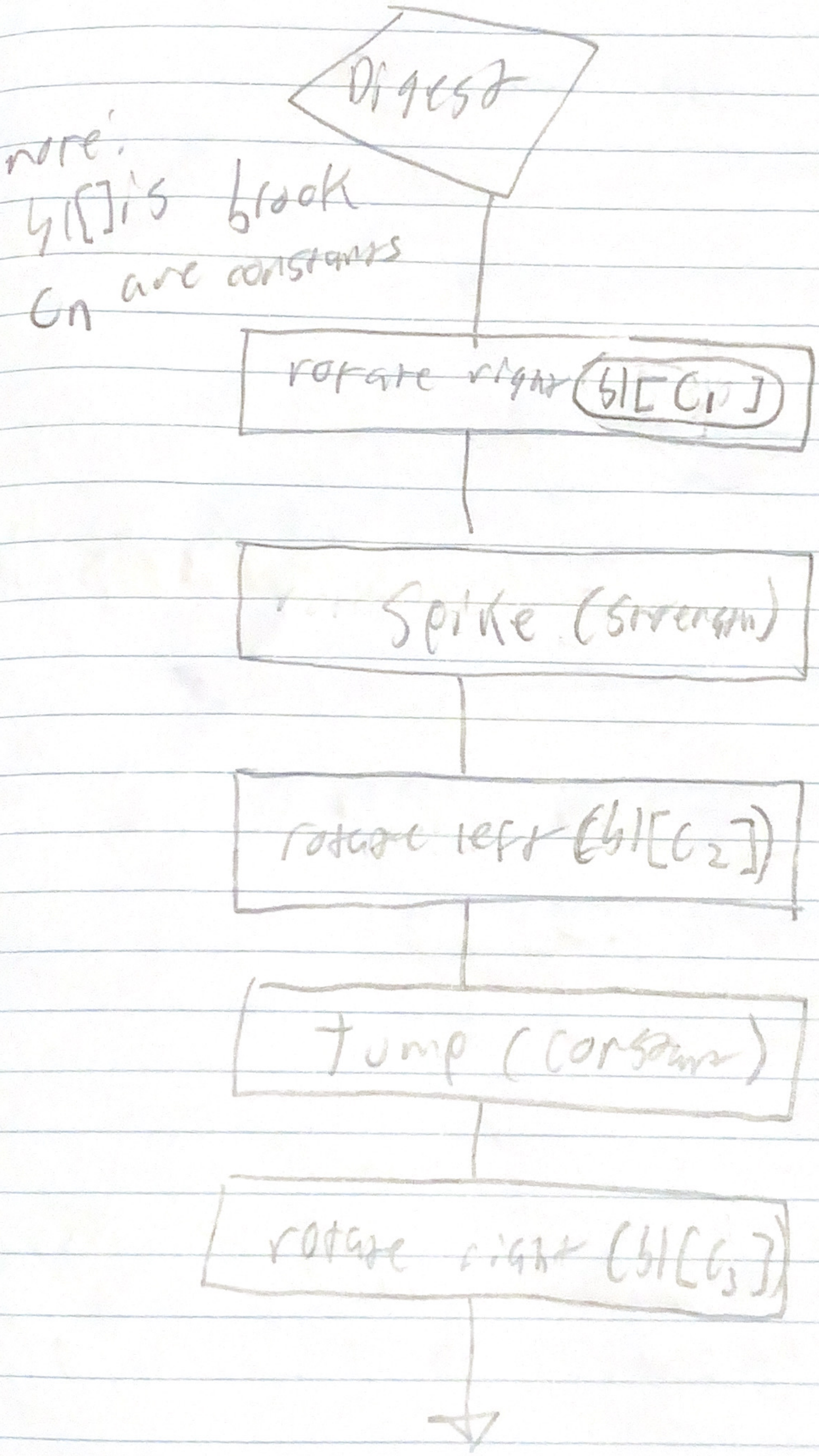
note! $A \vee B = \sim(A \wedge B)$

$C_a = 255 \% (C * S_c)$

$M_2(A,B,C) = \overbrace{(\neg A \wedge B)}^{T_1} \oplus \overbrace{(A \wedge C)}^{T_2} \oplus \overbrace{(B \vee C_a)}^{T_3} \odot \overbrace{(A \oplus B)}^{T_4}$

$\oplus \underbrace{(B \wedge C)}_{T_5}$

Block Seeding : Flowcharts



ACH-1
Nathaniel Aquino
10-16-21

[illegible]

[illegible]

- Functions should be [illegible] with sub blocks
  1:2

In the case of the block size of 1024 bytes, and with currently these funcs:

↳ Byte Spike (a)

↳ Byte Jump (a)

↳ OTP rotate [illegible] (a, 5)

↳

Sprite(a) = See notebook

Jump(a) = See notebook

Agate OTP(a, s)

Seeding constant based off the current computation iteration.

S((Sample A, Sample B, Sample C), Computation) =

Alternatively, sampling can be done in the C

S(Block[in], computation) =

where S_n are int representations of bytes at an index multiple of 340. (3 samples)

$$S(B,I) = f(I) = \sin\left(\frac{B_{s1} \cdot I}{20}\right) \cdot \cos\left(I^{B_{s2}}(B_{s3} \cdot I)\right)$$

$$f(i) = \frac{\sin\left(B[340] \cdot (i+10)\right)}{30} \cdot \cos\left((i+10)^{B[150]}\right)$$