Goldman Sachs | Bit Happens

Problem:

- Subtask 1: Decode a Base64 BMP, read the blue-channel LSBs, rebuild bytes, extract the number inside ABC{…}.
- Subtask 2: Decode a Base64 PCAP blob, treat it as text, regex-search for ABC{…}, take that big integer, compute (n%10007)+3 (or 0 if not found).
- Subtask 3: Decode Base64 hex bytecode, parse into opcodes, simulate the 8-bit MystVM until HALT, then return the byte at the given memory address.

Assumptions:

- If any decoding/parsing error occurs, that subtask's result defaults to 0.
- In case of multiple regex matches, only the first match is used.
- The Base64-decoded PCAP is already reassembled into a single text blob
- Java's BigInteger can handle the largest possible <number> in the input.
- Registers and memory initialize to zero; %r15 (SP) starts at 0.
- In case of multiple regex matches, only the first match is used.

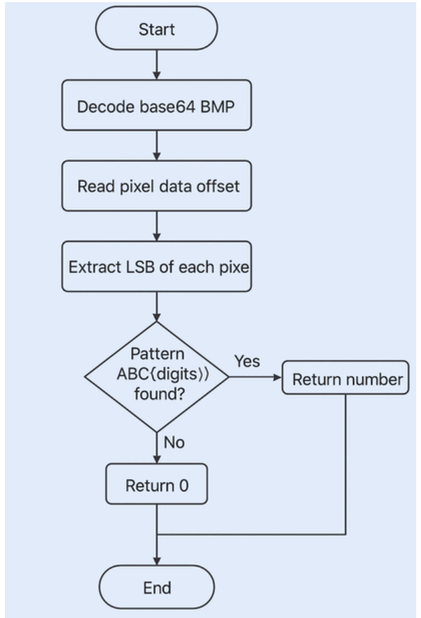
Subtask 1

Bitmap File Structure

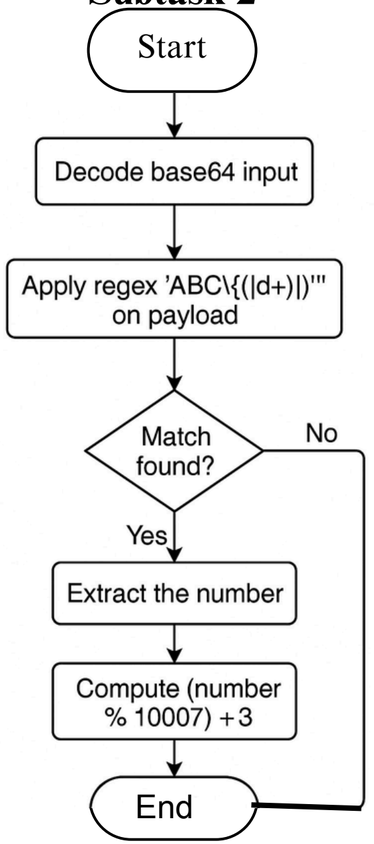
Block	Field	Width	Description
BITMAPFILEHEADER Fields: 5 Width: 14 bytes	FileType	2 bytes	A 2 character string value in ASCII. It must be 'BM' or '0x42 0x4D'
	FileSize	4 bytes	An integer (unsigned) representing entire file size in bytes (number of bytes in a BMP image file)
	Reserved	2 bytes	To be utilized by an image processing application. Initialized to '0' integer (unsigned) value.
	Reserved	2 bytes	To be utilized by an image processing application. Initialized to '0' integer (unsigned) value.
	PixelDataOffset	4 bytes	An integer (unsigned) representing the offset of actual pixel data in bytes.

Payload

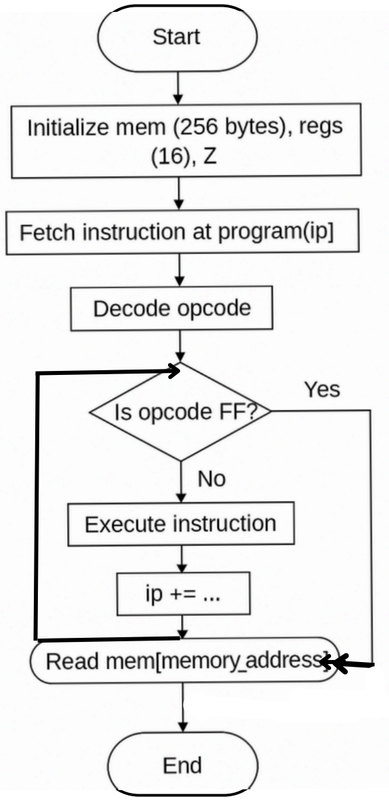
Each pixel: [Blue] [Green] [Red]
(Little Endian Format)



Subtask 2



Subtask 3



Why Java?

- Byte-level control & endianness via ByteBuffer and explicit masking (& 0xFF)
- High performance VM loops over millions of ops

Limitations & improvement

1. **No IP/TCP/UDP reassembly**

Curent fix: Applies regex on raw ISO-8859-1 decoded bytes
Future fix: Integrate a pcap library (Scapy/jNetPcap) to reassemble fragments before searching

2. **Stack pointer %r15 starts at 0, causing unintended underflow on first call**

Curent fix: Underflow wraps around silently via & 0xFF
Future fix: Initialize %r15 to 0xFF and add bounds checks (error on overflow/underflow).

3. **No early exit during BMP LSB scan**

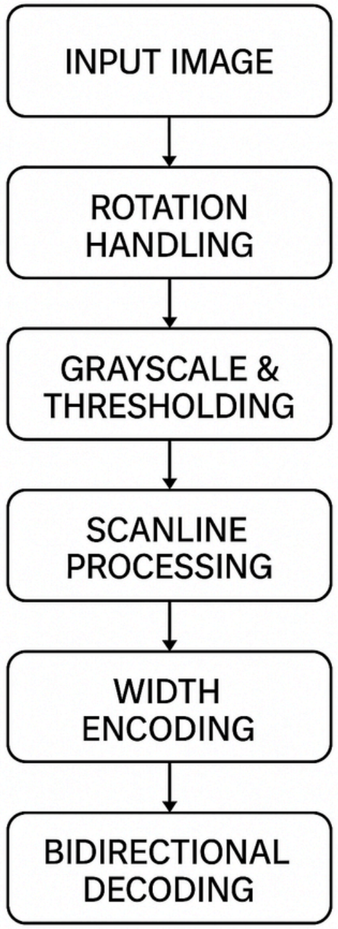
Curent fix: Collects full bitstream and reconstructs entire message
Future fix: Stop bit extraction as soon as regex match is found to save time

Goldman Sachs

Barcode Parsing

Problem: Decode a Code 39 barcode from a noisy, rotated image.
Handle real-world constraints: lighting, skew, inconsistent bar widths.

Approach:



Assumption: 1 barcode/image

Widths (9)	Pattern (N/W)	Char
[3,1,2,1,3,4,5,2,4]	NNNNNWWNW	G
[3,1,4,1,3,2,5,4,2]	NNWNNNWWN	S
[3,1,4,5,3,2,2,1,4]	NNWWNNNNW	2
[3,1,3,5,4,2,4,1,2]	NNNWWNWNN	0
[3,1,4,5,3,2,2,1,4]	NNWWNNNNW	2
[5,1,2,5,4,2,2,1,3]	WNNWWNNNN	5

Why Python?

- Fast prototyping, strong imaging libraries (PIL, NumPy).
- Ideal for iterative logic-heavy pixel processing.

Limitations & improvement

1. **Fails under extreme blur/skew**

   Curent fix: Otsu Thresholding
   Future fix: Deblurring filters or morphological edge enhancement

2. **Tilt beyond ±5° or vertical barcode**

   Curent fix: Small-angle rotations handle minor tilts.
   Future fix: Hough-line based deskew

3. **Corner-placed or tiny barcodes**

   Curent fix: We check ±3 rows around center
   Future fix: Sliding-Window + Early Pruning

Otsu's Thresholding Algorithm

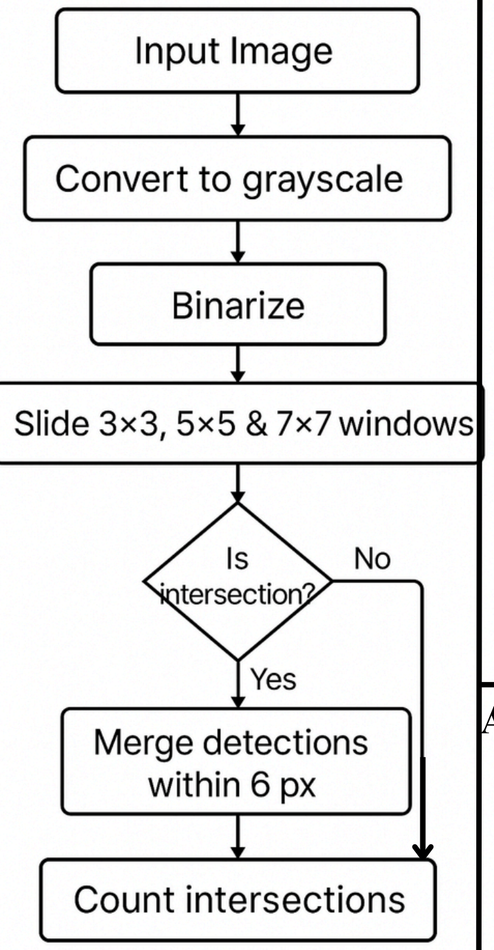
$$\sigma_B^2 = \frac{[m_G P_1(k) - m(k)]^2}{P_1(k)[1 - P_1(k)]}$$

Goldman Sachs

Automated Intersection point detection in 2D Line Graphs

Problem: Count line-line intersections in a 512×512 JPEG graph (axes excluded)

Approach:



3×3 ≥ 3

0	1	0
0	1	0
1	**1**	1

Ensures no noise

5×5 ≥ 6

0	0	1	0	0
0	0	1	0	0
1	1	**1**	1	1
0	0	0	0	0
0	0	0	0	0

Confirms the "cross" extends beyond 3×3

7×7 ≤ 30

0	0	0	1	1	0	0
0	0	0	1	1	0	0
0	0	0	1	1	0	0
1	1	1	**1**	1	1	1
0	0	0	1	1	0	0
0	0	0	1	1	0	0

Rejects thick blobs or axes (true 1px or 2px crossings only fill ~13–25 pixels in 7×7)

Limitations & improvement

1. **Heuristic Threshold Sensitivity**
Curent fix: Manually tuned as per 512x512
Future fix: Adaptive/local thresholding (Sauvola/Otsu)

2. **Over-Aggressive Clustering Radius**
Curent fix: 6 px merge radius to avoid duplicate detections.
Future fix: connected-component labeling and centroids or dynamic radius based on local line density.

3. **Axis & Origin Handling**
Curent fix: Implicitly ignored via 3 px border skip + 7×7≤30 rule.
Future fix: lightweight Hough transform or longest-line removal or padding.

Assumptions:

- Axes are thicker than data lines (≥ 3 px)
- No 2 valid intersection points occur within a 6px radius
- L pattern is not considered as an intersection

Alternative Solutions & Generalizations

Graph Skeleton & Node Analysis

- Morphologically thin binary image to a 1-px skeleton
- Treat skeleton as a graph: pixels = nodes, adjacencies = edges
- Identify graph nodes of degree ≥ 3 as intersections
- Scales to complex diagrams or network maps