# JPEG

1. לשים לב להבדל בין JPEG ל-JFIF ו-EXIF
2. לשים לב להבדלים בין JFIF ו-EXIF
3. כל APPn עבור n>2 כנראה מכיל מאטא דאטא
4. הסדר בין המרקרים באופן פרקטי אינו קבוע (לכאורה אמור כן להיות קבוע) אך כן מתחיל ב-APP0/1
5. DHT מכיל את טבלאות הופמן
   1. ~~אמור הגיונית להיות מחולק ל-3 חלקים פנימיים, אחד ל-Y אחד ל-Cb ואחד ל-Cr~~
   2. ~~פרקטית זה לא ככה (תמונות מאייפון ומגלקסי מכילות 4 חלקים), צריך לבדוק למה~~
   3. מכיל קידוד ל-AC וקידוד ל-DC לכל טבלה עתידית (לא בדיוק כי החיבור בין טבלה, קידוד ורכיב צבע יתבצע בעתיד ע"י SOF ו-SOS) ולכן מספרן אמור להיות זוגי ולרוב יהיה פי 2 ממספר טבלאות הקוונטיזציה.
   4. מבנה כל טבלת הופמן:
      1. מידע על הטבלה: בייט אחד המחולק כך:
         1. ביטים 0 עד 3: מספר הטבלה
         2. ביט 4 האם AC או DC (0 אומר DC)
         3. יתר הביטים צריכים להיות 0
      2. מספר הסמלים
      3. הסמלים עצמם
6. DQT מכיל את טבלאות הקוונציזציה
   1. אמור להכיל 2 עד 3 טבלאות, אחת ל-Y ואחת ל-Cb וגם ל-Cr או אחת לכל אחד מהם בנפרד
   2. נראה שזה באמת ככה פרקטית (תמונות מאייפון ומגלקסי מכילות 2 טבלאות)
7. SOF – קיימים שני סוגים של SOF. SOF0 אומר שזו התמונה הבסיסית, SOF2 אומר שזה פרוגרסיב (יש עוד תמונות אחרי זה)
   1. מכיל percision שאמור להיות 8 תמיד (אין סיבה שמישהו יעשה משהו אחר) זה מתאר את כמות הביטים שמתארים פיקסל בכל "צבע"
   2. גובה ורוחב – ברור
   3. מספר רכיבי הצבע, אמור להיות 3 תמיד
   4. לכל רכיב מתואר:
      1. ID
      2. צורת החלוקה הפנימית שלו (כמה פיקסלים לוקחים יחד בקבוצה ועושים עליהם ממוצע) זה בייט אחד שמחולק לשניים, 4 ביטים עליונים מתארים רוחב ו4 תחתונים גובה. זה עלול לעבוד בצורה טיפה מוזרה, זה לכאורה יחסי בינהם והפוך (אם כתוב שאחד הרכיבים הוא 2 הוא מדויק פי 2 מרכיב שיהיה כתוב לו 1)
         1. <https://stackoverflow.com/questions/43225439/jpeg-sof0-subsampling-components>
      3. מספר טבלת הקוונציזציה בה נשתמש עבור רכיב זה
8. SOS – "ה-DATA עצמו". מכיל:
   1. מספר הרכיבים (לרוב 3)
   2. לכל רכיב:
      1. ID
      2. מספרי טבלאות הופמן עבור AC וDC. הן נמצאות יחד על בייט אחד, DC בביתים העליונים
   3. עוד שלושה בתים לא ברורים. נראה שהערכים שלהם קבועים ואפשר לדלג עליהם, לא בטוח. צריך לבדוק.
   4. לא מכיל את ה-DATA עצמו (מבחינת SIZE), ה-DATA חייב לבוא מיד אחריו.
9. פה יש מלא מידע דחוס שצריך לפתוח לפי המידע שיש לנו קודם
10. EOI – סיום התמונה, באופן מפתיע עלול להיות מידע אחרי זה, אבל זה רק מאטא דאטא.

זה מאוד נוח:



וזה נחמד להתחלה ולא יותר מזה:



זה גם ממש אחלה:

<https://www.impulseadventure.com/photo/jpeg-huffman-coding.html>

פענוח MCU:

פענוח MCU:

בהתחלה יש את הDC של הY.

מקודדים לפי העץ, ומגיעים לערך. נסמן אותו x.

לוקחים x ביטים מסוף הערך, להם נקרא z.

מסתכלים על השורה של x בטבלה מספר 5, ושם יש 2 חלקים תחת additional bit

אם ה-MSB של z הוא 0, מסתכלים על צד שמאל של החלוקות ל-2, ואחרת על צד ימין. עבור x=3, למשל, הערך z הוא בגודל 3 ביט. אם ה-MSB שלו הוא 1, אז ליתר z יש 4 אפשרויות, שלפי הערכים DC VALUES יכולים להיות בין 4 ל7, ואז זה מחשבים בצורה מגעילה כשמתעלמים מהMSB, ואז את מה שנשאר מחברים ל"מינימום" (4). כנל רק הפוך עבור שליליים.

אחרי זה עושים את הAC של Y.

חשוב: הDC VALUE שערכו 0 משמעו EOB – סוף קידוד. אחריו הכל אפס. בדוגמה באתר, זה בא ישר. פה אין את המשחק של טבלה 5.

אחרי ה-Y עושים את ה-Cb, קודם DC ואז AC. אם יש EOB ב- DC אז גם מפסיקים (זה גם הגיוני וגם מסתדר עם טבלה 5).

מחשבה: האם אפשר לייצר אוטומטית את טבלה 5?

עושים את זה MCU אחרי MCU, ואחריהם יש ignore bits

אחרי ההמרה של הכל,