Robert Bielas, Michał Ćwiertnia
Zaawansowane systemy cyfrowe - projekt.
Dokumentacja.

Temat projektu:
Dekompresja plików graficznych w formacie jpg.

Cel:
Celem projektu jest zaprogramowanie zestawu uruchomieniowego DE-2 w ten sposób, aby mając na wejściu plik w formacie jpg, dokonał jego dekompresji, a wynik wyświetlił na wyświetlaczu VGA.

Założenia:
Wejście jest zahardkodowanym sprzętowo ciągiem bajtów opisującym zdjęcie w formacie jpg.
Projekt obsługuje tryb base-line kompresji stratnej wykorzystującej Discrete Cosine Transform(DCT) - najbardziej popularnej w Internecie.

Struktura jpg.
Format jpg jest zdefiniowany poprzez zbiór headerów(popularnie zwanymi markerami), z których każdy ma określoną składnię, znaczenie, pola i wartości określonych pól.
Markery mają sturkturę dwubajtową, z których pierwszy - ff - jest znacznikiem oznaczającym początek markeru, drugi zaś rodzaj markera.
Każdy plik rozpoczyna się poprzez marker początku pliku kóry ma wartość 0xffd8. Plik kończy się markerem 0xffd9.
Po znaczniku początku pliku występuje obszar metadanych. Jest to lista markerów o znaczeniu na poły informacyjnym, na poły opisowym. Najważniejsze dla nas są markery opisowe, które wymieniają najważniejsze charakterystyki obrazka, w tym jego długość, szerokość w pikselach, ilość komponentów graficznych, próbkowanie itp.
Tabelka poniżej pokazuje znaczenie poszczególnych znaczników.
Najważniejsze z punktu widzenia naszego projektu to xffc0, xffc4, xffda, xffdb.

xffc0 - marker który opisuje tryb wybrany w naszym projekcie.
Jego składnia: 2 bajty wskazujące długość tego markera (bez symbolu xffc0, razem z 2 bajtami opisującymi długość), bajt precyzji, 2 bajty ilości rzędów, 2 bajty ilości pikseli w rzędzie, bajt ilości komponentów (N) , a potem N 3 bajtowych opisów komponentów (id, samplowanie, klasa Huffmana).

xffdb - opis tabel(ek/ki) kwantyzacji wykorzystywanych przy dekompresji.
Składnia:
2 bajty opisujących długość M, ilość tabelek wyliczamy N = (M - 2) /(64 + 1)
N tabelek:
1 bajt, z którego pierwsze 4 bity to precyzja (0 - precyzja 8bitowa, 1- precyzja 12 bitowa, to zazwyczaj jest jednak 0), następne 4 bity opisujące id tabelki.
Następnie występują 64 bajty właściwej tabelki zapisanej zygzakiem.

xffc4 - znacznik tabelki Huffmana
2 bajtowa długość M,
1 bajt - pierwsze 4 bity komponentu używającego tabelki, następne 4 to klasa drzewa(0 - opisujące współczynniki DC, 1 - opisujące współczynniki AC).

następnie N tabelek, gdzie

$$N = \frac{M-2}{\sum_{i=1}^{16} len(i) + 1}$$

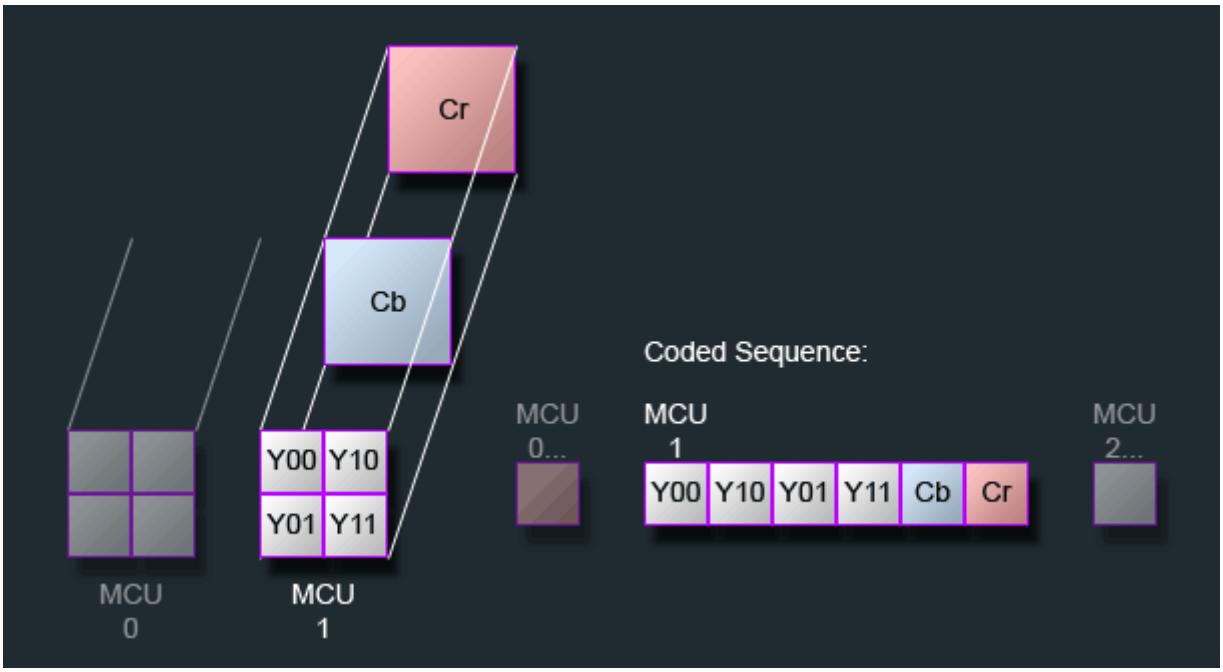
len to ilość kodów i bitowych

xffda - start of scan
składnia: 2 bajtowe długość N
Następnie 1 bajtowa ilość komponentów M i 3 bajtowe opisy tych komponentów w postaci:
pierwszy bajt - id komponentu,
drugi bajt - pierwsze 4 bity to id tabelki huffmana dla klasy 0(dc), następne 4 bity to id tabelki huffmana dla klasy 1(ac)
Następnie 3 bajty które są stałe dla naszego rodzaju kompresji.
Całość treści po tych znacznikach to już właściwy opis kolorów w pliku, zakończone ewentualnym dopełnieniem wymiarów pliku do rozmiarów będących wielokrotnością 8 i obowiązkowym znacznikiem 0xffd9 końca pliku.

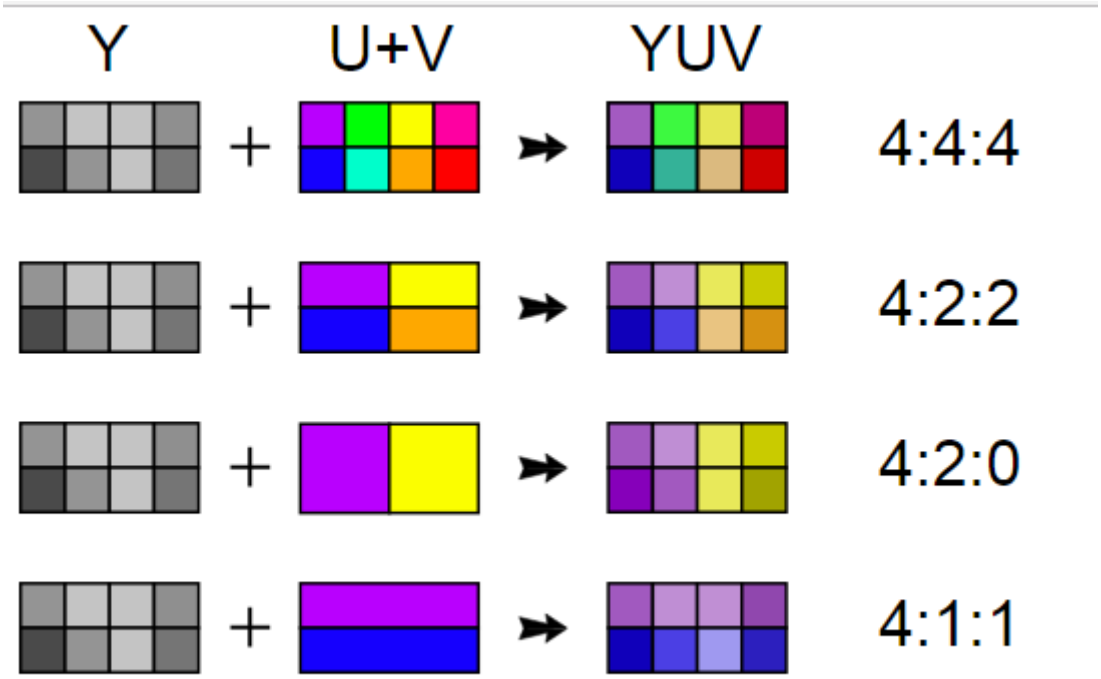
Code Assignment	Symbol	Description
Start Of Frame markers, non-differential, Huffman coding		
X'FFC0' X'FFC1' X'FFC2' X'FFC3'	SOF_0 SOF_1 SOF_2 SOF_3	Baseline DCT Extended sequential DCT Progressive DCT Lossless (sequential)
Start Of Frame markers, differential, Huffman coding		
X'FFC5' X'FFC6' X'FFC7'	SOF_5 SOF_6 SOF_7	Differential sequential DCT Differential progressive DCT Differential lossless (sequential)
Start Of Frame markers, non-differential, arithmetic coding		
X'FFC8' X'FFC9' X'FFCA' X'FFCB'	JPG SOF_9 SOF_{10} SOF_{11}	Reserved for JPEG extensions Extended sequential DCT Progressive DCT Lossless (sequential)
Start Of Frame markers, differential, arithmetic coding		
X'FFCD' X'FFCE' X'FFCF'	SOF_{13} SOF_{14} SOF_{15}	Differential sequential DCT Differential progressive DCT Differential lossless (sequential)
Huffman table specification		
X'FFC4'	DHT	Define Huffman table(s)
Arithmetic coding conditioning specification		
X'FFCC'	DAC	Define arithmetic coding conditioning(s)
Restart interval termination		
X'FFD0' through X'FFD7'	RST_m*	Restart with modulo 8 count "m"
Other markers		
X'FFD8' X'FFD9' X'FFDA' X'FFDB' X'FFDC' X'FFDD' X'FFDE' X'FFDF' X'FFE0' through X'FFEF' X'FFF0' through X'FFFD' X'FFFE'	SOI* EOI* SOS DQT DNL DRI DHP EXP APP_n JPG_n COM	Start of image End of image Start of scan Define quantization table(s) Define number of lines Define restart interval Define hierarchical progression Expand reference component(s) Reserved for application segments Reserved for JPEG extensions Comment
Reserved markers		
X'FF01' X'FF02' through X'FFBF'	TEM* RES	For temporary private use in arithmetic coding Reserved

Opis procesu. Etapy dekompresji.
Dekompresja jest procesem symetrycznie odwrotnym do procesu kompresji jpg. Aby odtworzyć wartości skompresowane w pliku formatu jpg, przede wszystkim należy rozszyfrować wartości zakodowane w strumieniu następującym po znaczniku xffda.
Obrazek jest koncepcyjnie podzielony na bloki 8x8 pikseli. Jest to najbardziej podstawowa jednostka podziału, z którego budowane są bardziej złożone struktury.
Omawianie tych struktur należy zacząć od omówienia procesu próbkowania.
Głównym celem jpga jest zmniejszenie rozmiarów obrazka korzystającego z przestrzeni kolorów RGB i wszystkie zabiegi, algorytmy oraz mechanizmy w tym formacie są tworzone właśnie z myślą o tym. Inżynierowie zajmujący się tym zagadnieniem wykorzystali zjawisko charakterystyczne dla ludzkiego oka: największy wkład w postrzeganie przez nas świata mają czopki, które są znacznie bardziej czułe na zmianę intensywności światła aniżeli sam kolor. Za odbiór kolorów w naszym oku odpowiedzialne są zdecydowanie mniej wrażliwe pręciki.
Pomysł polega na tym, aby przestać używać 3 bajtowej przestrzeni kolorów RGB i wykorzystać bardziej oszczędną, lecz równoważną przestrzeń YCbCr. odpowiadającą temu zjawisku. Y jest elementem(komponentem) odpowiadającym za reprezentację natężenia światła, tzw. luminacja. Cb i Cr to komponenty chrominancji (domieszki koloru) odpowiednio niebieskiego i czerwonego koloru. Domieszka zieleni nie jest potrzebna, gdyż kolor zielony w przestrzeni RGB jest obliczalny przy użyciu jedynie tych trzech powyższych wartości.

Wykorzystując tą przestrzeń, wprowadza się proces próbkowania. Ogólnie mówiąc na kilka komponentów Y przypada jeden lub dwa elementy chrominancji.
W naszym parserze obsługujemy najbardziej powszechne próbkowanie 4:2:0, co oznacza, że na 4 wartości komponentu Y(ułożone w kwadrat 4 bloki 8x8 pikseli) przypada 1 komponent Cr i 1 komponent Cb.



Taki układ możemy zgrupować w nieco bardziej złożoną strukturę zwaną MCU (minimal coded unit).



Obrazek: Wkład poszczególnych komponentów w przestrzeni YCbCr w różnych trybach próbkowania.

Każdy z komponentów MCU to blok 8x8 pikseli. Taki układ nie jest niczym przypadkowym, gdyż dzięki temu możemy te komponenty poddać na wejście powszechnie używanego w kopmresji formatu jpg przekształcenia macierzowego jakim jest dyskretna transformata kosinusowa, którą definiujemy w następujący sposób:

$$U = \frac{1}{2}\begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \cos\frac{\pi}{16} & \cos\frac{3\pi}{16} & \cos\frac{5\pi}{16} & \cos\frac{7\pi}{16} & \cos\frac{9\pi}{16} & \cos\frac{11\pi}{16} & \cos\frac{13\pi}{16} & \cos\frac{15\pi}{16} \\ \cos\frac{2\pi}{16} & \cos\frac{6\pi}{16} & \cos\frac{10\pi}{16} & \cos\frac{14\pi}{16} & \cos\frac{18\pi}{16} & \cos\frac{22\pi}{16} & \cos\frac{26\pi}{16} & \cos\frac{30\pi}{16} \\ \cos\frac{3\pi}{16} & \cos\frac{9\pi}{16} & \cos\frac{15\pi}{16} & \cos\frac{21\pi}{16} & \cos\frac{27\pi}{16} & \cos\frac{33\pi}{16} & \cos\frac{39\pi}{16} & \cos\frac{45\pi}{16} \\ \cos\frac{4\pi}{16} & \cos\frac{12\pi}{16} & \cos\frac{20\pi}{16} & \cos\frac{28\pi}{16} & \cos\frac{36\pi}{16} & \cos\frac{44\pi}{16} & \cos\frac{52\pi}{16} & \cos\frac{60\pi}{16} \\ \cos\frac{5\pi}{16} & \cos\frac{15\pi}{16} & \cos\frac{25\pi}{16} & \cos\frac{35\pi}{16} & \cos\frac{45\pi}{16} & \cos\frac{55\pi}{16} & \cos\frac{65\pi}{16} & \cos\frac{75\pi}{16} \\ \cos\frac{6\pi}{16} & \cos\frac{18\pi}{16} & \cos\frac{30\pi}{16} & \cos\frac{42\pi}{16} & \cos\frac{54\pi}{16} & \cos\frac{66\pi}{16} & \cos\frac{78\pi}{16} & \cos\frac{90\pi}{16} \\ \cos\frac{7\pi}{16} & \cos\frac{21\pi}{16} & \cos\frac{35\pi}{16} & \cos\frac{49\pi}{16} & \cos\frac{63\pi}{16} & \cos\frac{77\pi}{16} & \cos\frac{91\pi}{16} & \cos\frac{105\pi}{16} \end{bmatrix}$$

Elements of the 8 x 8 DCT matrix.

Zastosowanie tego narzędzia to kolejny przykład ukierunkowania twórców formatu na maksymalną kompresję pliku graficznego. Ma ono bardzo przydatną właściwość: piksele w bloku zostają przetransformowane z domeny przestrzennej do domeny częstotliwościowej o takiej własności, że najbardziej dominująca częstotliwość jest sprowadzana do lewego górnego rogu macierzy (Czyli elementu nr 1 macierzy), zaś reszta to generalnie wartości równe zero lub bliskie zeru.

Pierwszy element nazywamy współczynnikiem DC dyskretnej transformaty kosinusowej(direct current), pozostałe 63 to współczynniki AC.
Poniżej wizualizacja wyniku DCT:

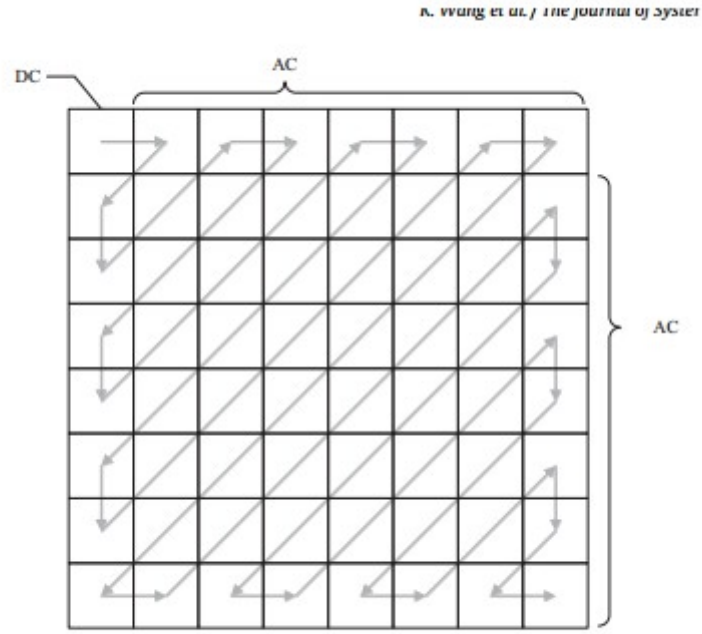


Fig. 3. Zigzag sequence.

Strzałkami oznaczono w jaki sposób części komponentów są kodowane od postaci dwuwymiarowej macierzy do jednowymiarowego strumienia danych - tzw. zigzag order.
Mając na uwadze informacje dot. układu komponentów w trybie próbkowania 4:2:0 w pojedynczym MCU, należy przejść do etapu kodowania, jakim jest tzw. entropy coding. Najbardziej popularnym (niekomercyjnym i intuicyjnym) tego typu kodowaniem jest kodowanie Huffmana. W baseline DCT przy użyciu tego szyfru kodujemy komponenty następująco: pierwszy element (dc) bloku komponentu jest kodowany przy użyciu drzewa huffmana klasy 0.
Kolejne 63 elementy są opisane przez sekwnecję hc1 hv1 ... hcn hvn 0x00.
hci to skrót od huffman code o indeksie i, hvi oznacza huffman value o indeksie i, dotyczące drzewa huffmana klasy 1(ac). Koniec bloku współczynników ac jest oznaczane bajtem EOB(end of block) o wartości 0x00. Jeżeli jesteśmy na indeksie mniejszym niż 63 gdy występuje EOB, to pozostałe elementy bloku wypełniamy zerami.
Wartości hvi to jednobajtowe kody, w których: pierwsze 4 bity oznaczają ilość kolejnych elementów komponentu będącymi zerami(RZL - run zero length), zaś następne 4 bity wartość długości w bitach n już właściwej wartości v, która następuje w macierzy komponentu po tej sekwencji zer, a która jest zakodowana dalej w strumieniu. Dla przypomnienia wszystkie elementy bloku są zapisywane w porządku zig zag.
Ta wartość v jest kolejnym rodzajem kodu: bezpośrednią wartość elementu komponentu v' możemy odczytać z poniższej tabelki w zależności od wartości n i v i ją właśnie wpisać po sekwencji RZL.
Uwaga: element dc odszyfrowujemy korzystając z tej samej tabelki, lecz siłą rzeczy pierwsze 4 bity hv będą tam zawsze wyzerowane.

DC Code	Size	Additional Bits		DC Value	
00	0			0	
01	1	0	1	-1	1
02	2	00,01	10,11	-3,-2	2,3
03	3	000,001,010,011	100,101,110,111	-7,-6,-5,-4	4,5,6,7
04	4	0000,...,0111	1000,...,1111	-15,...,-8	8,...,15
05	5	0 0000,...	...,1 1111	-31,...,-16	16,...,31
06	6	00 0000,...	...,11 1111	-63,...,-32	32,...,63
07	7	000 0000,...	...,111 1111	-127,...,-64	64,...,127
08	8	0000 0000,...	...,1111 1111	-255,...,-128	128,...,255
09	9	0 0000 0000,...	...,1 1111 1111	-511,...,-256	256,...,511
0A	10	00 0000 0000,...	...,11 1111 1111	-1023,...,-512	512,...,1023
0B	11	000 0000 0000,...	...,111 1111 1111	-2047,...,-1024	1024,...,2047

Bezpośrednio po odczytaniu jawnych wartości współczynników dc i ac z powyższej tabelki należy użyć tzw. tabeli kwantyzacji. Zazwyczaj są dwie: jedna odpowiada elementom komponentu luminancji, zaś druga elementom Cr i Cb. Przy kompresji współczynniki dzieli się przez odpowiednie elementy w tabelce kwantyzacji (dobranej tak, by parametry obrazka były optymalne), zaś przy dekompresji - wymnaża się je przez siebie.

Zdekodowane współczynniki DC w komponentach nie są jednak jeszcze ostateczne, tylko zapisane jako różnica w stosunku do elementu dc poprzedniego komponentu odpowiedniego rodzaju w porządku w jakim pojawiły się w zakodowanym strumieniu. Tylko pierwsze komponenty każdego rodzaju mają w strumieniu zakodowaną bezwzględną wartość elemtnu dc (czyli od nich nic nie odejmujemy).

Wzory do przejścia z przestrzeni YCbCr do przestrzeni RGB są następujące:
Red = Y + 1.402 * Cr + 128
Green = Y - 0.3437 * Cb - 0.7143 * Cr + 128
Blue = Y + 1.772 * Cb + 128

Biorąc to wszystko pod uwagę, dekompresja (po sparsowaniu metadaty po odpowiednich markerach, odczytaniu specyfikacji tabelek huffmana i zbudowaniu w oparciu o nie drzew, oraz zainstalowaniu wszystkich tabel kwantyzacji) polega więc kolejno na:
1. Zidentyfikowaniu rodzaju próbkowania - domyślnie 4:2:0;
2. dla każdego koponentu:
    - odczytać współczynnik dc przy użycia drzewa klasy 0 i powyższej tabelki, wpisać go do komponentu
    - odczytać sekwencję hc1 hv1 .. hcn hvn 0x00 i w opisany wyżej sposób zdekodować i wpisać do komponentu, używając porządku zig-zag.
    - dla każdego komponentu C użyć tabelki kwantyzacji Q o odpowiednim id(domyślnie - id tabelki kwantyzacji komponentu luminancji wynosi 0, dla komponentu chrominancji 1) i wymnożyć je przez siebie tak, że powstaje nowa macierz C' której elementy C'(i)=C(i)*Q(i).
3. Odczytać bezwzględne wartości DC wszystkich komponentów we wszystkich MCU.
4. Dokonać Inverse DCT, czyli przejścia z domeny częstotliwościowej do domeny przestrzennej.
5. Przejść z przestrzeni YCbCr do RGB dla każdego komponentu we wszystkich MCU.
6. Utworzyć trzy tabele R',G' i B' o rozmiarach odczytanych po znaczniku 0xffc0
7. Ułożyć elementy RGB w każdym MCU w odpowiadającym im miejscach w tablicach R'G' i B'
8. dostosować wartości w tabelach R' G' i B' tak by były zawarte w przedziale (0,255)