1. **P Čemu kod MPEG kodera služi povratna veza nakon entropijskog kodera natrag prema kvantizacijskom modulu?`**

*Povratna veza nakon entropijskog kôdera (VLC ili ENC) služi za određivanje koraka*

1. **Obavlja li se pretvorba prostora boja RGB u YUV (YCrCb) prije ili nakon postupka kompresije? Zašto?**

*Pretvorba prostora boja se obavlja prije kompresije, kako bi se mogla razdvojiti svjetlina i boja, odnosno luminantna i krominantna komponenta slike.*

*Kada imamo te komponente razdvojene onda većom kompresijom kodiramo komponentu koja prikazuje boje (krominantnu komponentu) na koje je oko manje osjetljivo nego na komponentu svjetline (luminaciju) na koju je oko osjetljivije*

*Skraćeno: Izvodi se prije postupka kompresije kako bi se podaci koji nose informacije o boji mogli više komprimirati od luminantne komponente*

1. **Za sliku veličine 1024×768 piksela odrediti broj operacija množenja i zbrajanja (zbrajanja i oduzimanja) u slučaju pretvorbe iz RGB u YUV prostor pomoću sljedećih formula:**

**Y = (0.257\*R) + (0.504\*G) + (0.098\*B) + 16;**

**U = -(0.148\*R) - (0.291\*G) + (0.439\*B) + 128;**

**V = (0.439\*R) - (0.368\*G) - (0.071\*B) + 128;**

*Za svaki piksel se mora obaviti 9 mnozenja i 9 zbrajanja. 1024\*768 = 786432 piksela*

*Dakle 1024\*768\*9 zbrajanja i 1024\*768\*9 mnozenja.*

1. **Odredite broj operacija množenja (množenje i dijeljenje) i zbrajanja (zbrajanje i oduzimanje) pri "full search" metodi procjene pokreta na jednom bloku 8×8 pri čemu je područje pretraživanja +/-8 piksela u obje dimenzije, a mjera poremećaja MAD (predviđanje se obavlja na Y komponenti).f**

Za blok 8x8:

* 64 oduzimanja (zbrajanja) i 1 množenje (dijeljenje s 64)
* // što nije 64 oduzimanja i 63 zbrajanja kada računamo MAD ?? i to je onda 127 zbrajanja
* //Kolegi iznad, slajd 58 MAS3 skoro isti zadatak samo 16x16...pa kaze 256 oduzimanja i 1 pomak (dijeljenje s 256)
* // hvala, a jel moze netko objasnit zasto uzimaju u obzir samo oduzimanje a ne i zbrajanje ?
* // Nema to veze sa zdravim razumom. Za izračun MAD-a jednostavno se mora uzeti u obzir i zbrajanja (63). Tko izračuna MAD bez zbrajanja, plaćam pivu.

Pomak +-8 piksela u obje dimenzije:

* (8\*2+1)\*(8\*2+1)\*(64+dohvat) = 18496 zbrajanja + dohvati
* (8\*2+1)\*(8\*2+1)\*(1+dohvat) ◼ Za blok 16x16: 256 oduzimanja + 1 pomak (dijeljenje= 289 množenja + dohvati
* //Što predstavlja ovaj drugi point - 289 množenja + dohvati? U slajdovima se ne spominje ovo. //Tako sam nasao u rjesenjima zadatka sa Multimedijskih Tehnologija bio je identican zadatak..

1. **Navesti barem tri algoritma za brzu procjenu pokreta.**

* Algoritmi za brzu procjenu pokreta:
* *Logaritamsko pretraživanje (LOG) //TDL*
* *Pretrazivanje u 3 koraka (3ss) // TSS*
* *Ortogonalno pretrazivanje (ORT) // OSA*
* *Algoritam gradijentnog spusta temeljen na blokovima (BBGDS)*

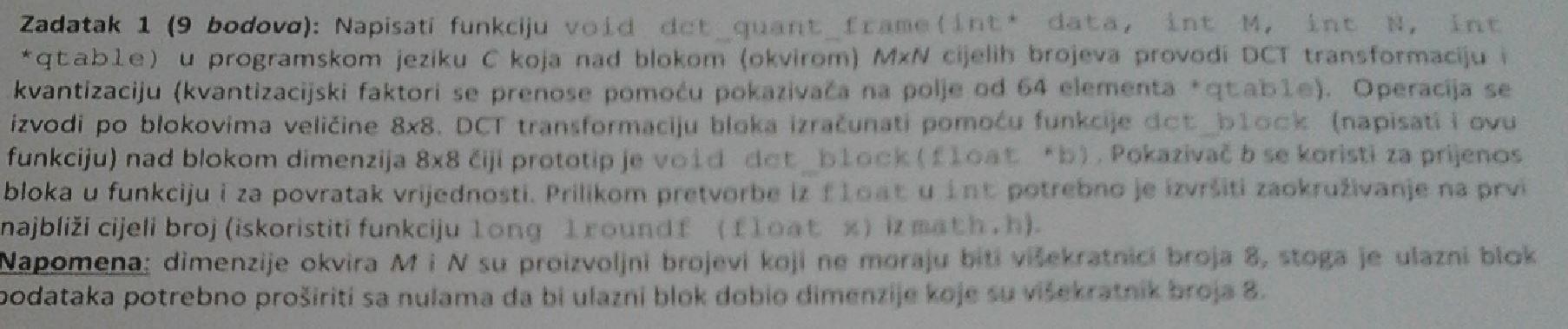
1. **Objasniti zašto se u JPEG normi koriste različiti kvantizacijski koeficijenti za luminantnu i krominantnu komponentu slike.**

*Zbog veće osjetljivosti oka na luminantnu komponentu (svjetlinu) nego na krominantnu komponentu (boje)*

1. **Koji su nedostaci brzih algoritama za procjenu pokreta?**

*Ne daju uvijek najbolji rezultat, jedino kod potpunog pretrazivanja smo sigurni da je stvarno nađen najsličniji blok*

1. **DCT transformacija: Zadani su prototipovi dviju funkcija. Potrebno je implementirati obje funkcije u programskom jeziku C (kompletni program). Prva funkcija kao argumente prima pokazivač na niz podataka \*data, veličinu polja podataka N i M te pokazivač na kvantizacijsku tablicu \*qtable veličine 8\*8. Potrebno je nad danim podacima napraviti DCT transformaciju i kvantizaciju. Druga funkcija prima samo pointer na 8 x 8 blok podataka i ona radi DCT transformaciju nad predanim joj blokom, transformirane vrijednosti vraća preko istog polja podataka.**



**void\_quant\_frame(int\* data, int M, int N, int \*qtable){**

**float blok[8][8];**

**while(M % 8 != 0){M++;}**

**while(N %8 != 0){N++;}**

**for(m=0; m<M; m=m+8)**

**for(n=0; n<N; n=n+8)**

**for(int i=0; i<8; i++)**

**for(int j=0; j<8; j++)**

**blok[i][j] = data[m+i][n+j];**

**dct\_block(blok);**

**for(int i=0; i<8; i++)**

**for(int j=0; j<8; j++)**

**data[m+i][n+j] = lroundof(blok[i][j] / qtable[i][j]);**

**}**

**void dct\_block(float \*b){**

**float f[8][8] = {0};**

**float a;**

**for(int i=0; i<8; i++){**

**for(int j=0; j<8; j++){**

**f[i][j] = b[i][j];**

**}**

**}**

**for (int i = 0; i < 8; i++) {**

**for (int j = 0; j < 8; j++){**

**a = 0.0;**

**for (int x = 0; x < 8; x++) {**

**for (int y = 0; y < 8; y++) {**

**a += (float)f[x][y] \* cos((2.0\*(float)x + 1.0)\*(float)i\*3.14 / 16.0)\*cos((2.0\*(float)y + 1.0)\*(float)j\*3.14 / 16.0);**

**}**

**b[i][j] = (0.25\*Calculate(i)\*Calculate(j)\*a);}**

**}**

**}**

**return;**

**}**

**float Calculate(int i){**

**if (i == 0) {**

**return (1.0 / sqrt(2.0));**

**}else {**

**return 1.0;**

**}**

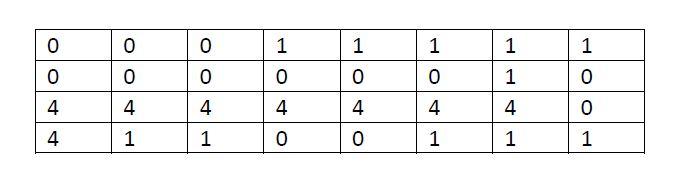
**}**

**//jebo ih C  
 // ^**

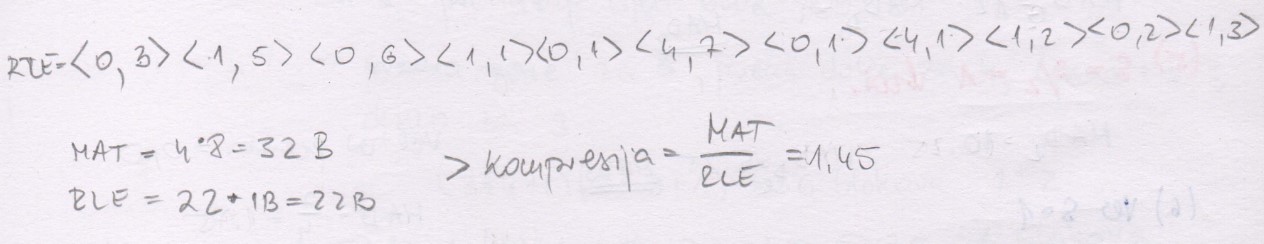
**//Mozda je netko u zadaci rijesio ovo pa nek iskopira kod..**

**//Nema nekih značajnih fora ja mislim, samo treba paziti da imaš jedan pokazivač za više manje sve, a želiš da budu kao 2d polja, pa onda imaš (npr za b) b[širina\*i+j], tj u našem slučaju b[8\*i+j]**

1. **Zadan je blok podataka:**

****

**Potrebno je podatke kodirati RLE kodom u obliku <S,N> pri čemu je S simbol a n broj uzastopnog ponavljanja simbola S. Odrediti stupanj kompresije ovakvog koda.**

****//otkud 22\*1B? - kod RLE <S,N> svaki S je 1B i svaki N je 1B, u ovom slučaju je naći 11 S i 11 N što nam daje ukupno 22B

**//** kad bi npr. nule bile na kraju retka i na početku sljedećeg jel ih sve pišemo u jedne zagrade <,> ili treba raz **??????? netko?  
Ja bih rekla da se sve pise u jedne zagrade, jer u koder ulazi kao stream?  
+ gledala sam kako bi se to iskodiralo pa sam naisla na ovo  
“The easiest way to do 2D RLE encoding is to take the input signal, reshape it into a vector, and do 1D RLE encoding on the vector, giving an output”**

1. **Video rezolucija- vidi prezentaciju MAS3,4 slajd**

**Jedan okvir je veličine 1920 \* 1080, RGB format, 8 bita po komponenti. U sekundi se šalje 24 okvira. Odrediti brzinu nekompresiranih podataka. Kakav mora biti stupanj kompresije ako želimo da brzina komprimiranih podataka bude 5MB/s ?**

*1920\*1080\*24\*3 B(komponenta RGB-a je 1B) = 149299200 B / (1024\*1024) = 142.38 MB/s*

*142.38/5 MB/s / MB/s = 28.47 -> stupanj kompresije*

*// Bilješka: Vrijednost K (Kilo) u izračunima može imati dvije vrijednosti 1024 ili 1000, ovisno koju vrstu izračuna želite obaviti. Razmislite o korištenju K = 1024, kada je u pitanju kapacitet za pohranu bilo na tvrdi disk, DVD, flash pogon ili na druge uređaje i medije za pohranu. K = 1000 se obično koristi kada se misli na propusnost, odnosno brzinu kojom se informacije prenose.*

*Dakle ne dijeli se sa (1024x1024) već sa 10^6 … tako je riješeno i na prez. MAS3, 4 slajd*

1. **Navesti i pojasniti metode dodjele sabirnica u višeprocesorskim sustavima.**

*T – traženje sabirnice (BREQ – Bus Request)*

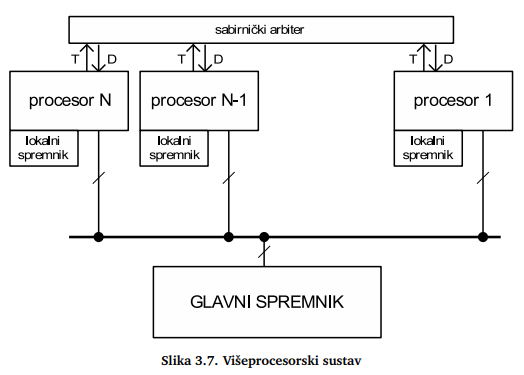
*D – dodjela sabirnice (BACK – Bus Acknowledge)   
• sabirnicu dijele svi procesori   
• lokalni (priručni) spremnik se koristi da se smanji potreba za sabirnicom*

*//Nije odgovor na pitanje…*

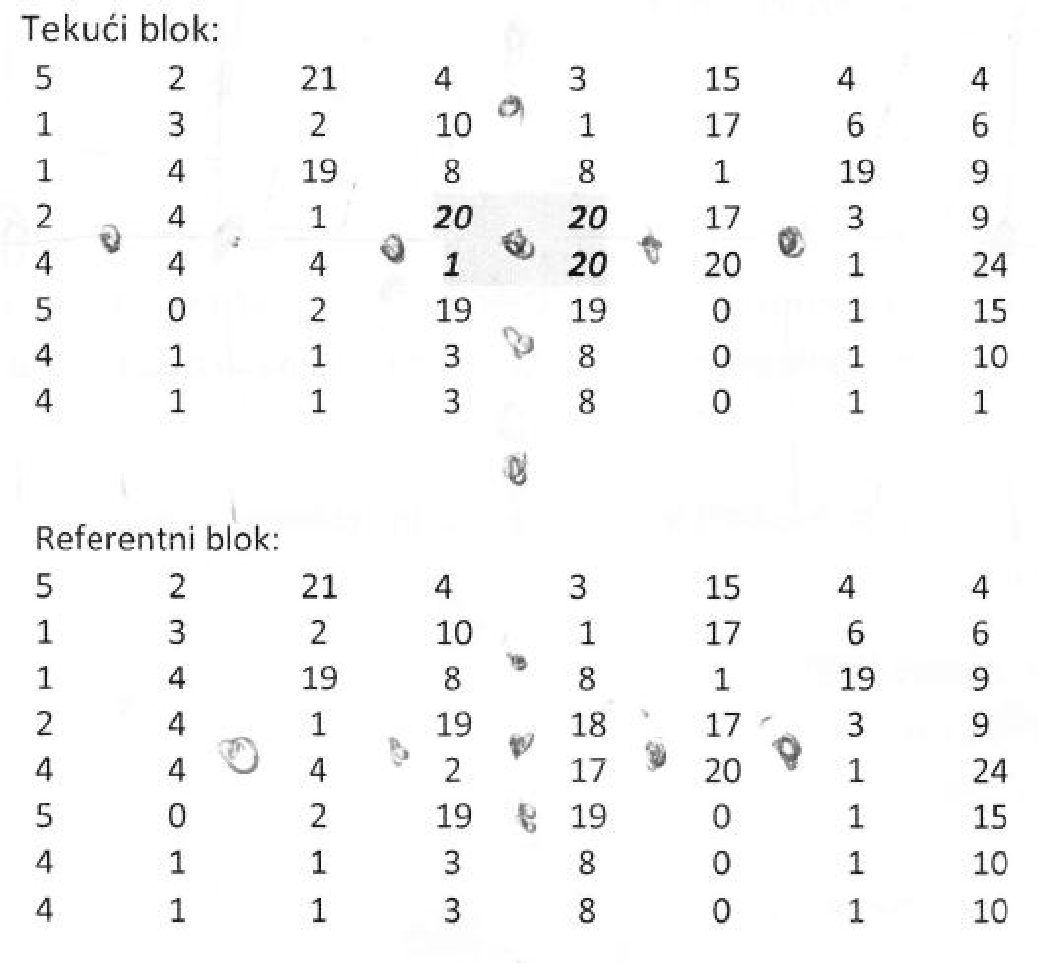
*// koje nije odgovor? ovo ispod?*

*Kod višeprocesorskih sustava, svaki procesor ima svoju cache memoriju i sabirnicu.*

*Kod višejezgrenih sustava sve jezgre dijele zajedničku cache memoriju i sabirnicu*

**

U višeprocesorskim sustavima postoji dodatni sklop – sabirnički arbitar, koji omogućuje dijeljenje sabirnice među procesorima, tako da se ne dogodi konflikt pri korištenju sabirnice. Sabirnički arbitar upravlja sabirnicom po načelu zahtjev/odgovor. Kada procesor treba koristiti sabirnicu, on najprije traži dozvolu od arbitra signalom traži (T, engl. bus request). Kada na njega dođe red, arbitar mu dodjeljuje sabirnicu i o tome ga obavještava signalom dodjela (D, engl. bus acknowledge).

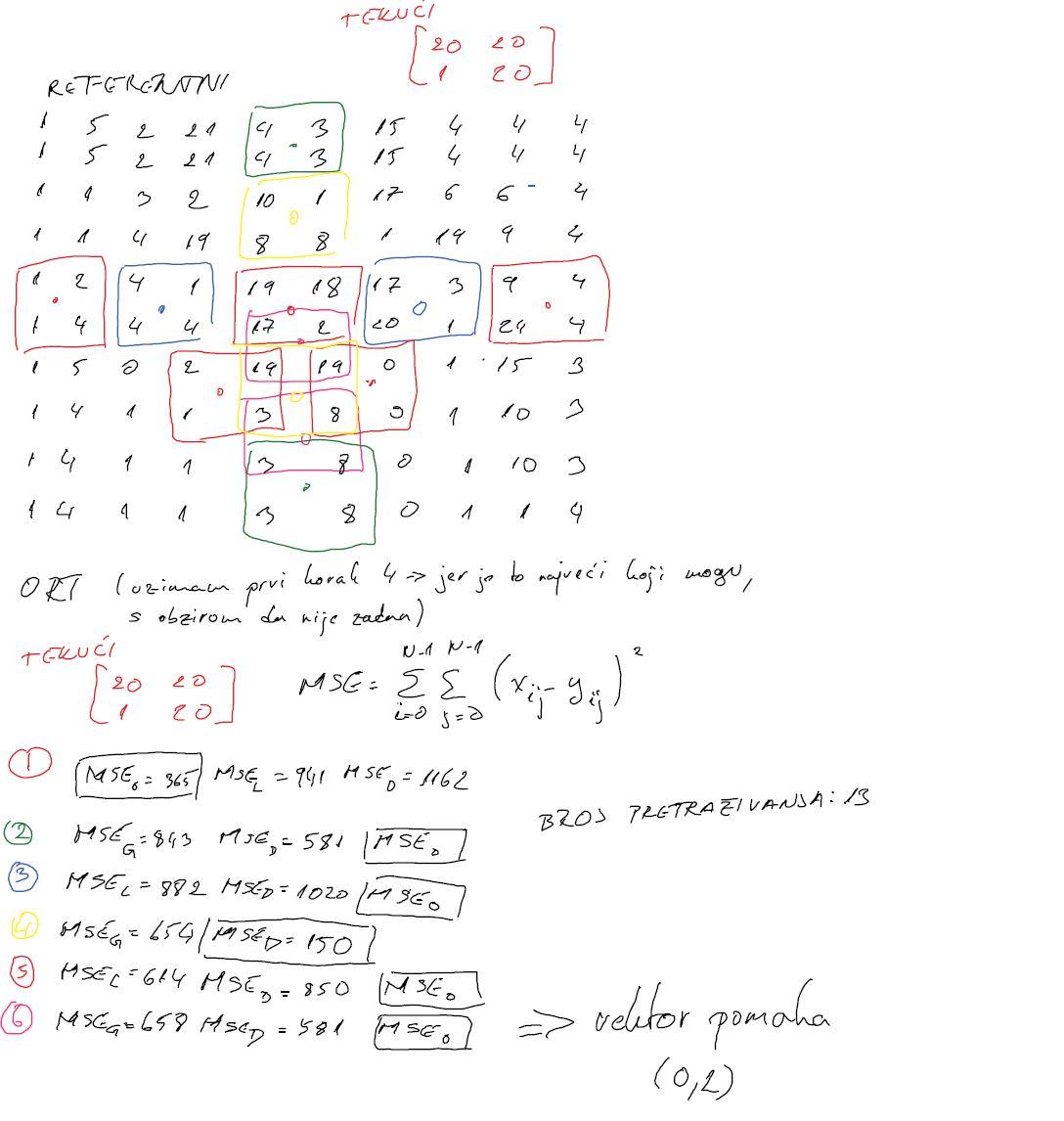
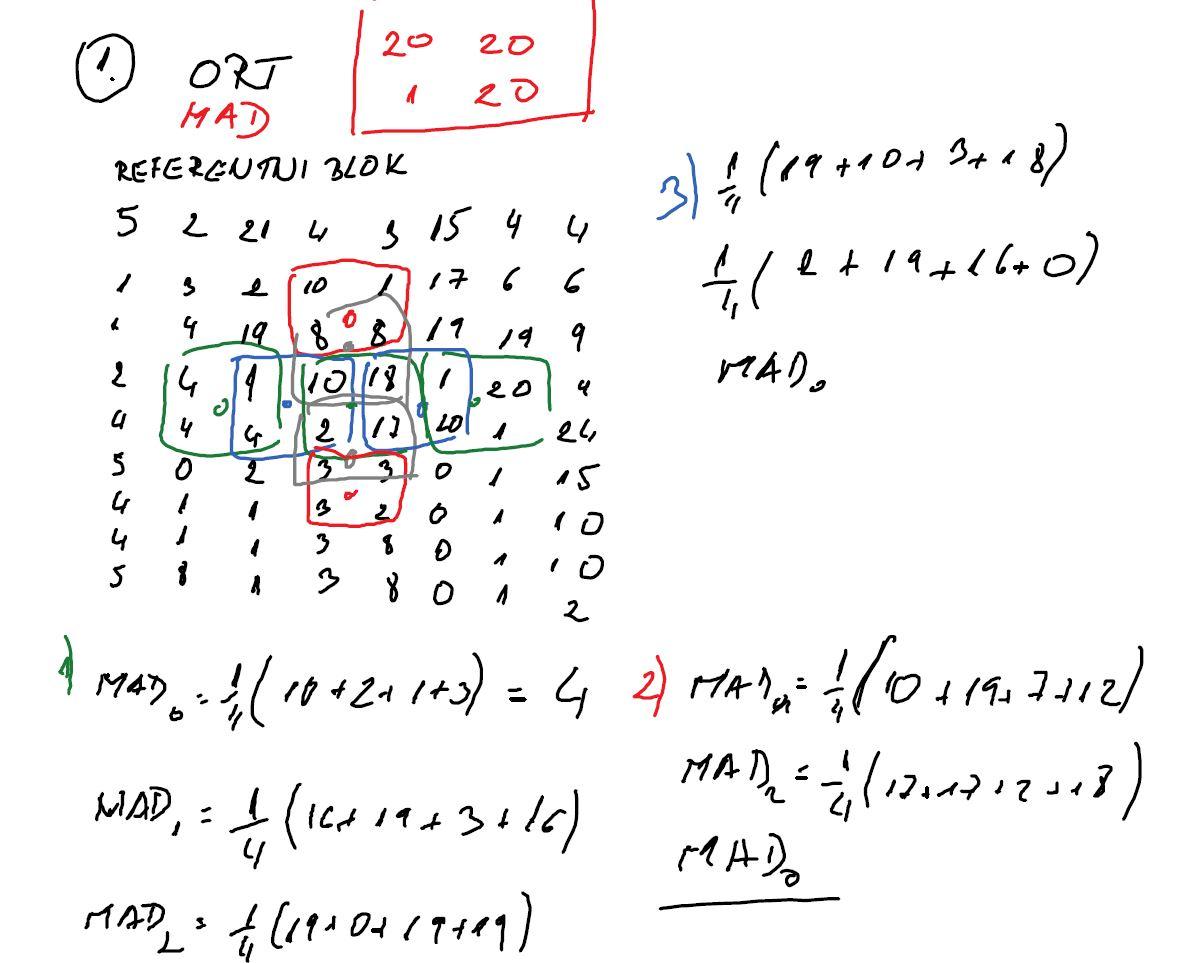


**SLIKA**

1. **SLIKA 1: Definirati broj potrebnih operacija zbrajanja i mnozenja za izračun vektora pomaka iz zadatka 1. Usporediti broj operacija sa brojem operacija algoritma potpunog pretrazivanja pri cemu je velicina prozora 6 slikovnih elemenata.**

*3SS*

* *za s=3 računamo MAD 9 puta*
* *za s=2 računamo MAD 8 puta*
* *za s=1 računamo MAD 8 puta*
* *9+8+8 = 25 izračuna, a svaki put smo radili 3 zbrajanja, 4 oduzimanja i 1 množenje tj.*
* *25\*3 (+)*
* *25\*4 (-)*
* *25\*1 (\*)*

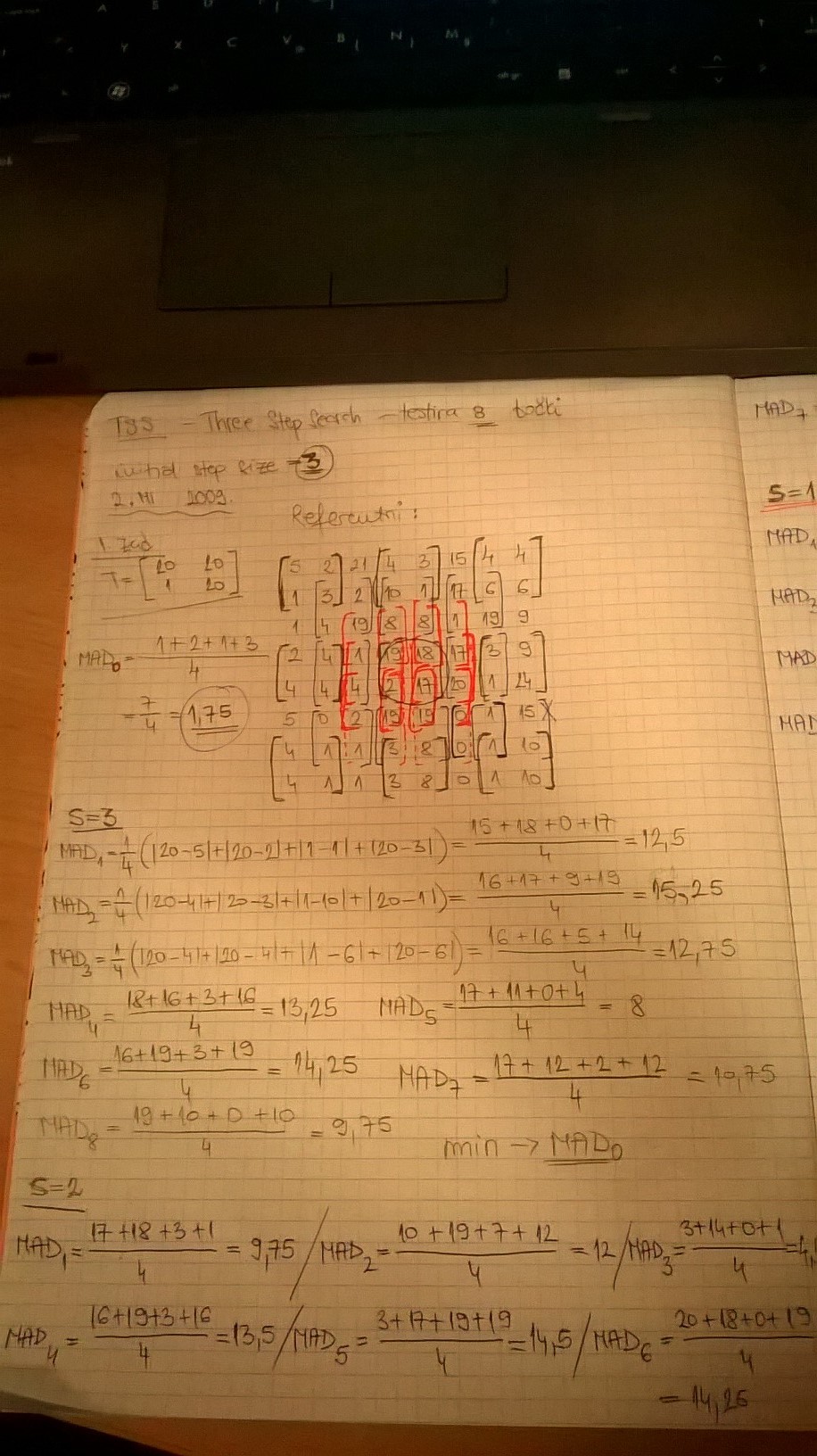
**Slika 1 :Određivanje vektora pomaka ME postupkom. Zadan je ORT algoritam s početnim korakom . Tekući i referentni blok ispod. (isti podaci kao 2MI 2009/2010, ove oznake na slici su za drugi algoritam - tada je bio zadan 3SS, zanemariti)**

*Potrebno objasnit slike malo bolje…*

*//jel se na kraju zna koliki je pocetni korak na pocetku algoritma, odnosno kako se definira? Ovdje pise da je s=4, zasto je poceo sa korakom 2?*

***\*\*\*\*\* POGLEDATI KRAJ DOC-a \*\*\*\****

*//tu ispod su riješena mislim dva različita zadatka, za ovaj lijevi je s=2, a za desni je s=4, obrati pažnju i da piše tekući i referentni okvir ispod za s=4// u svim ovim zadacima je srednji ispao najbolji, što ako nije takav slučaj? //Piše ti za svaki algoritam (prezentacija kompresija videa), ako je neki drugi novo središte pretraživanja, pomičeš se na njega i oko njega tražiš dalje ako sam dobro shvatio*

* **SLIKA 1 - Pretpostada gornja 2 bloka podataka predstavljaju ulazne slike u video koder. Ako pretpostavimo da je veličina bloka 2x2 i ako pretpostavimo algoritam pretrazivanja 3SS (Three Step Search), izračunati vektor pomaka za označeni blok u gornjem okviru. Koristiti MAD kao mjeru poremecaja.**
* *oduzimanje je zapravo zbrajanje komplementa pa dobivamo 25\*7 (+) i 25\*1(\*)*
* *175 zbrajanja i 25 množenja*

*// nije uračunat dohvat svakog bloka, tako da vjv. treba dodati i 25 dohvata*

*Full search*

* *6 elemenata znači da MAD = ⅙ \* (|a-b|+|c-d|+|e-f|+|g-h|+|i-j|+|k-l|) tj.*
* *6 (-)*
* *5(+)*
* *1(\*)*
* *oduzimanje je zapravo zbrajanje pa imamo 11(+) i 1(\*) za svaki blok*
* *u zadatku imamo 8x8 brojeva, ako blok ima 6 brojeva znači da taj blok može proći po mreži od 8x8 brojeva 7 pozicija desno i 6 pozicija dolje tj. ukupno 7\*6=42*
* *462 zbrajanja i 42 množenja*
* *znači da ćemo 42 puta računat MAD koji ima 11(+) i 1(\*) tj.*

1. **Objasniti prostornu, spektralnu i vremensku redundanciju (bilo na MI 2018.)**

**Prostorna -** vrijednosti piksela u manjim područjima vrlo su slične

**Spektralna -** kad se podaci preslikavaju u frekvencijsku domenu, nekoliko frekvencija dominira nad drugima

**Vremenska -** baš kao što su vrijednost piksela lokalno slične unutar jedne slike, one su također povezane i između slikovnih okvira, npr u videu se većina piksela koji čine pozadinu ne razlikuje mnogo od okvira do okvira. Područja koja se mijenjaju su ili objekti koji se pomiču ili se pomiče kamera, ali ti su pokreti kontinuirani i stoga predvidljivi

1. **Objasniti zašto se za Y komponentu koristi različita kvantizacijska tablica u odnosu na U i V komponente.**

*?Jer je na Y komponentu oko najosjetljivije pa se nju slabije komprimira nego U i V ?*

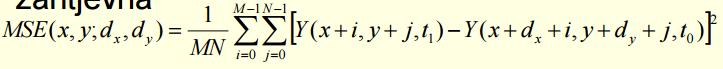
*Kvantizacijski koeficijenti za U i V biti će veći zbog HVS*

Kao što je kolega gore napisao, s obzirom na veći broj štapića u ljudskom oku nego čunjića, on je osjetljiviji na Y, koja je luminantna komponenta. Cr i Cb predstavljaju krominantnu komponentu, stoga je potrebno imati dva različita skupa kvantizacijskih vrijenosti za dvije različite komponente, kako bi se kompresija napravila kak se spada, bez gubitka bitnih podataka.

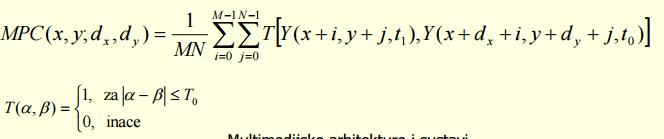
1. **Objasniti zašto se vrijednosti slikovnih elemenata umanjuju za (n je preciznost elemenata) prije ulaza u blok za DCT transformaciju.**

Umanjuju se zato jer je DCT područje signed vrijednosti. Odnosno, za pixele preciznosti 8 bita odnosno [0, 255] moraju se umanjiti za 2^(8-1) = 128, da se dobije **signed raspon.**

1. **Definirati mjere poremećaja pri izračunu vektora pokreta**

**

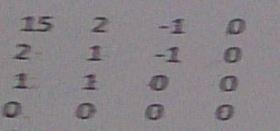
*2.JPG*

**

*//može netko malo pojasniti kako se točno računa ova treća mjera?*

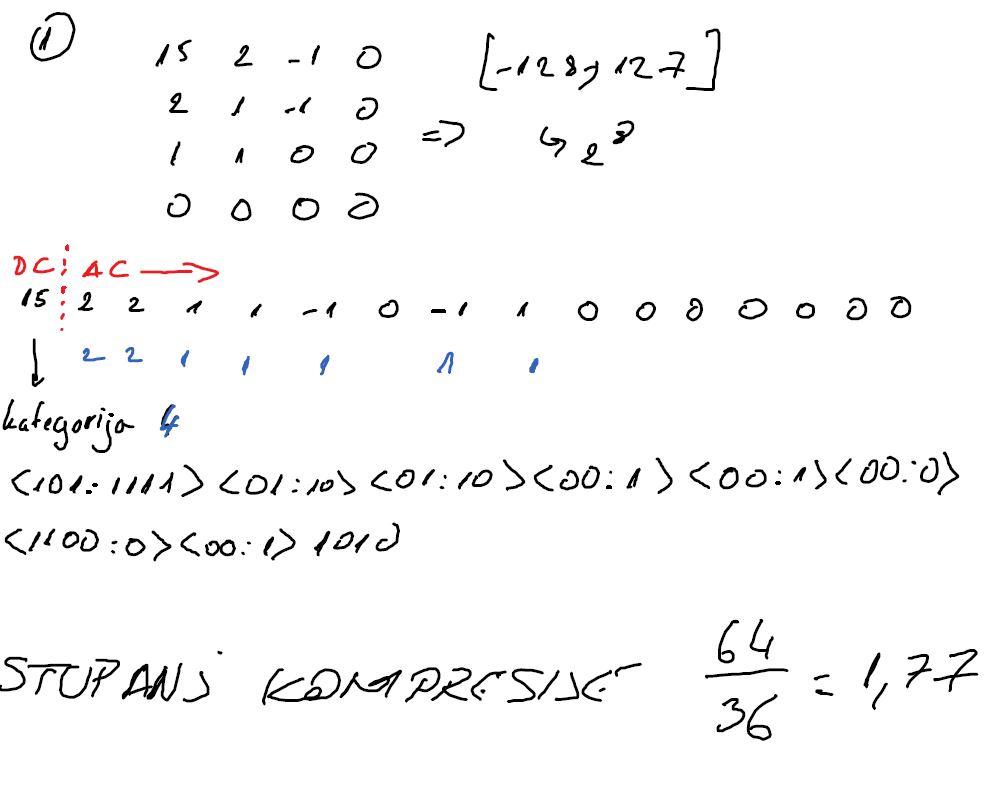
*//Ako je apsolutna razlika elementa danog i usporednog bloka manja ili jednaka nekom zadanom pragu (threshold theta), onda je count=1, inače je 0. I ti countovi se samo pozbroje, kao što se za MAD zbrajala apsolutna razlika. I da, traži se maximum valjda (MPC = Matching Pixel Count)*

1. **Za zadani blok kvantiziranih DCT koeficijenata konstruirati zig-zag slijed koji ulazi u entropijski koder**

****

**Odrediti Huffmanov kod za dobiveni slijed simbola te stupanj kompresije ukoliko su ulazni simboli vrijednosti u rasponu od -128 do 127**

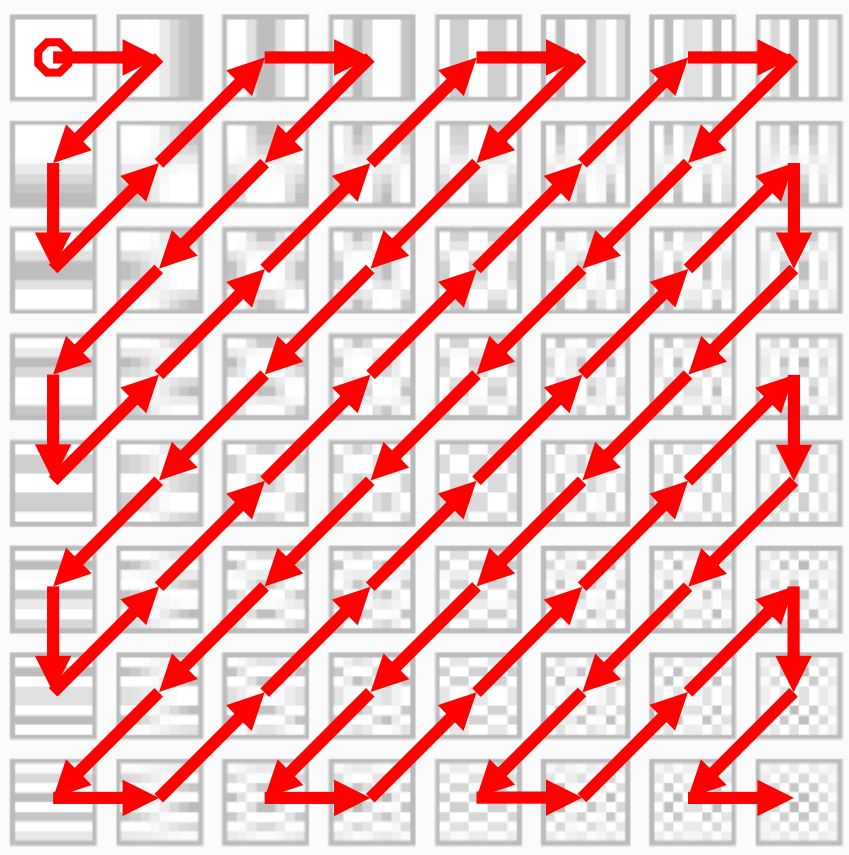
**DC**

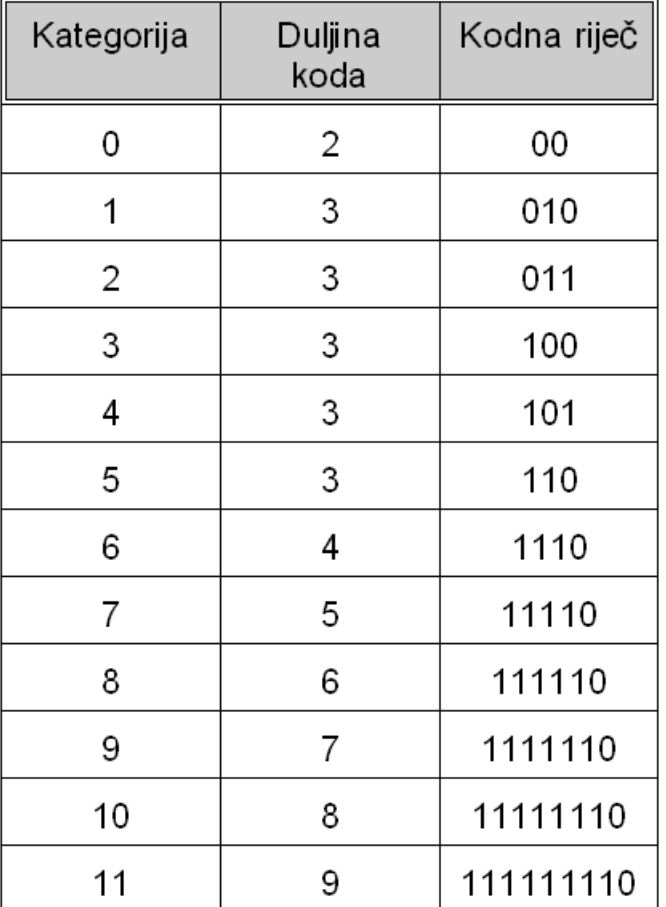
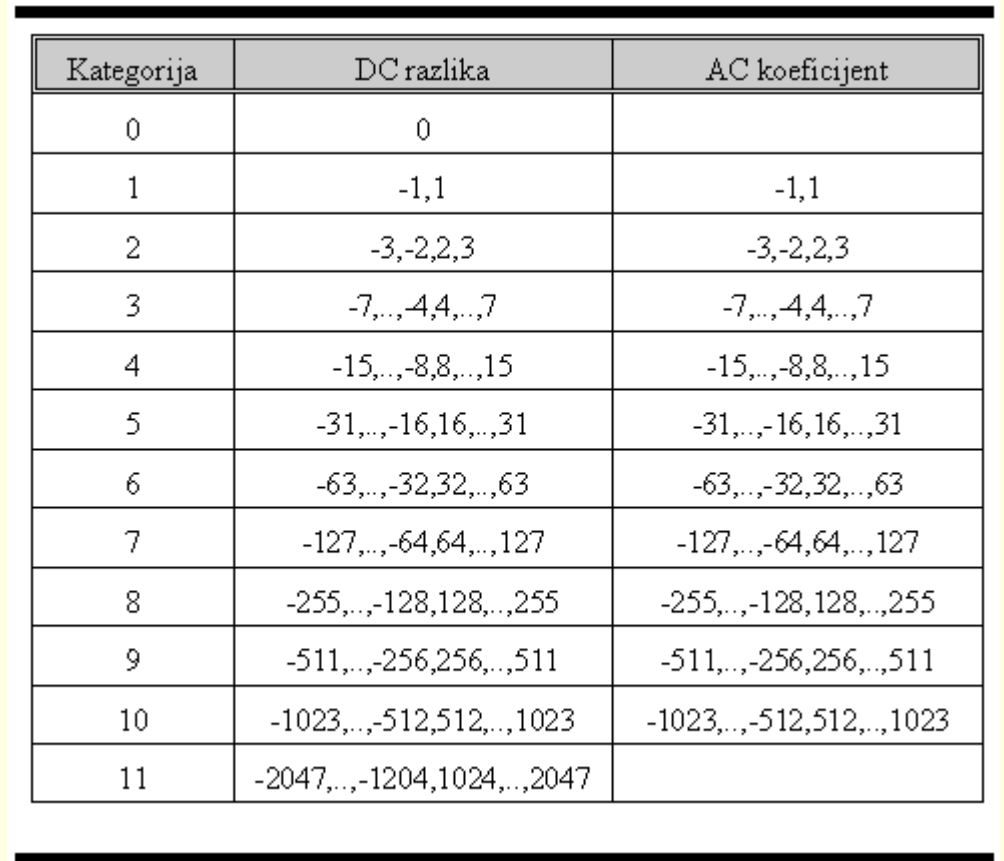


*<101:1111><01:10><01:10><00:1><00:1><00:0><1100:0><00:1><1010>*

*Pojasniti….*

*//kako se za originalnu dužinu dobilo 64? ako su ulazni simboli u vrijednosti od -128 do 127, //zar nije to 8 bita po simbolu? onda bi kompresija bila 128/36 = 3.556*

*// kako se računa amplituda za negativne brojeve?* 

*Potrebno gledati tablice u prezentaciji MAS 2 tamo negdje oko 79-ih slajdova… Koriste se za određivanje kodnih rijeci i kategorija*



Dakle po redu:

15,2,2,1,1,-1,0,-1,1,0,0,0,0,0,0,0  
Ovakav ispis se dobije kada idemo zig-zag po tablici

Prvi broj u nizu je uvijek DC, ostali su AC

zapisuje se na sljedeci nacin:

DC <kodna riječ kategorije : amplituda (binarni broj)>

,AC

Budući da nema nijednog drugog bloka 15 se ne mora od niceg oduzet

stoga je 15-0 = 15 i DC je 15 // slajd 70, 2. prezentacija i 20. slajd kompresija slika

15 spada prema tablici (DC) u kategoriju(broj bitova potrebnih za binarni zapis broja) 4 (jer se moze zapisati sa 4 bita), za kategoriju 4 kodna rijec je 101, Dakle slijedi:

15 -> <101,1111> …. 1111 je binarni zapis broja 15

Sada slijede AC brojevi. Duljina niza nula ISPRED tog broja se gleda i njegova kategorija te se trazi kodna rijec, dakle slijedi: (napomena negativni brojevi se racunaju tako da nadjemo binarni zapis pozitivnog broja i onda puknemo komplement i dobimo negativni)

2 -> <01,10> // 01 je zbog duljina niza nula/kategorija -> 0/2 i binarni zapis dvojke je 10

// moze neko objasnit oo gore? kako 01 zbog duljina niza/kategorija? duljina binarnog zapisa broja 2 je 2 i po tome kategorija treba biti 0?

//pogledaš samo da li ti se prije tog broja pojavljuje uzastopan broj 0, ako se pojavljuje, broj tih nula je prvi broj, a kategorija tog broja je drugi, gore u skroz lijevoj tablici pogledaš kod za prvi broj/drugi broj

2 -> <01,10>

1 -> <00,1> //0/1 ...0 nula ispred jedinice i kategorija 1 kodna rijec 00, binarni zapis je 1

1 -> <00,1> // 0/1 kodna rijec 00 i

-1 je komplement(1)->0

//zar se negativni brojevi ne zapisuju pomoću dvojnog komplementa, pa bi trebalo pribrojiti jos 1 na ovu nulu pa bi bilo : -1 -> <00,1> ??

//kolegi iznad, to ne bi imalo puno smisla jer bi onda -1 i 1 dali potpuno isti kod, zar ne?

0 -> Ne racunamo

-1 -> <1100:0> //Sada je kodna rijec 1100 jer je jedna nula ispred -1, kategorija 1 i onda je 1/1->1100, binarni zapis je 0 zbog komplement(1)

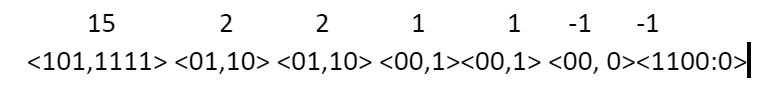
1 -> <00,1>

I jos 7 nula

1010 EOB(end of block)

Dakle dobijemo od 16 brojeva (128b) 36b kodiranih, kompresije 128/36 = 3.56.

Zapisi:



1. **Za 8K UHDTV vide u formatu 21:9 rezolucije 10080x4320 slikovnih elemenata, u RGB prikazu pri cemu je preciznost slikovnog elementa u jednoj komponenti 8 bita, te za 120 okvira u sekundi, izračunati potrebnu brzinu nekomprimiranog prijenosa. Dodatno , odrediti potreban stupanj kompresije za gornje podatke da bi se teoretski mogli slati putem 4G LTE bezicne mreze (maksimalna brzina preuzimanja je 37,45 MB/s)**

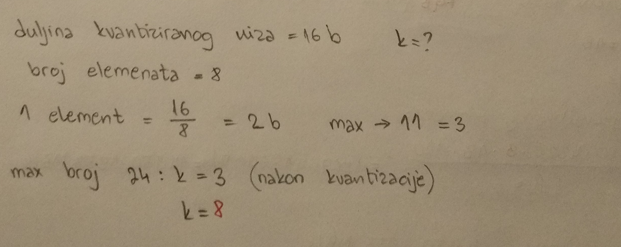
*10080\*4320\*120\*3 B = 15676416000 B / (1024\*1024) = 14950,19 MB*

*Potrebno je slati 14950,19 MB / s*

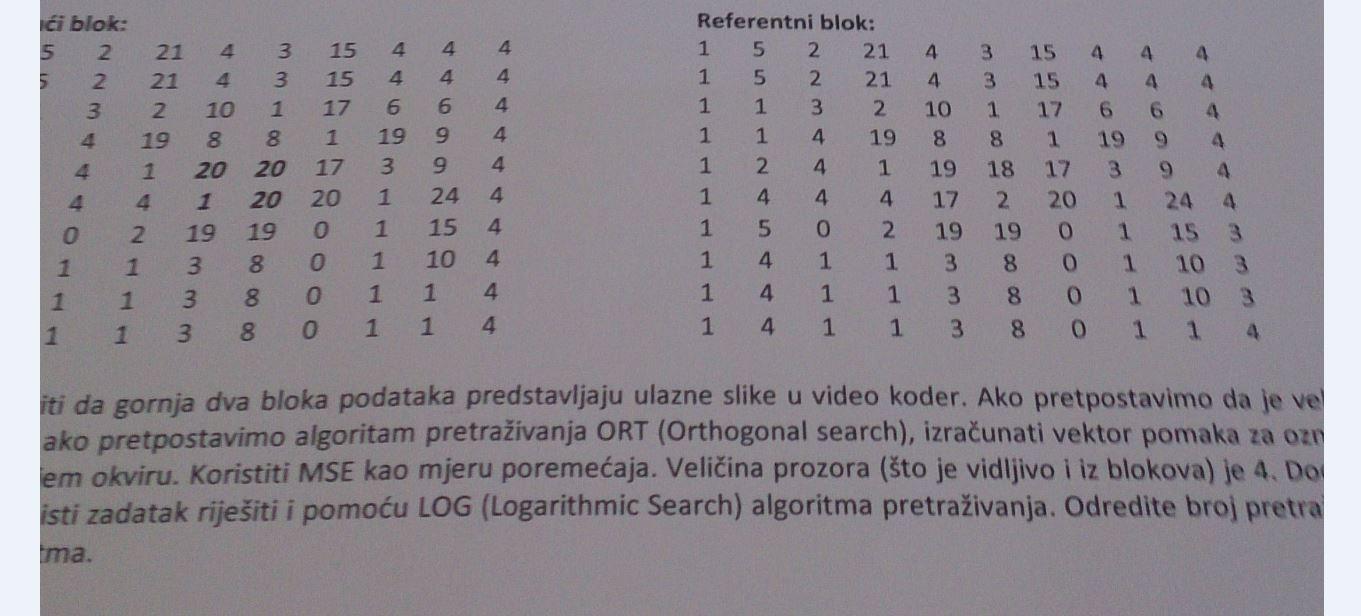
*14950,19 / 37,45 = 399,2 -> stupanj kompresije*

*Točno rješenje: 10080\*4320\*120\*3 B = 15676416000 B / 10^6 = 15676.416 MB/s  
Stupanj kompresije: 15676.416 / 37.45 = 418.5959*

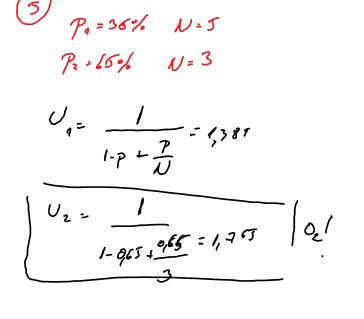
1. **Za postupak kvantizacije nad ulaznim nizom 0 , 5, 10, 16, 24, 4, 5, 8 uz kvantizacijski korak 5 odrediti:**
   1. **Koliko minimalno bitova potrebno za zapis ulaznog ne kvantiziranog niza?**
      1. *najveći broj je 24 koji se može zapisat u 5 bitova jer 2^5 = 32*
      2. *5 \* 8 = 40 bitova za čitav niz*
   2. **Koliko minimalno bitova potrebno za zapis izlaznog kvantiziranog niza q u slučaju zaokruživanja kvantizirane vrijednosti [q] i u slučaju zaokruživanja [q]?**
      1. *q = 5, slučaj ceil(q) -> 0,1,2,4,5,1,1,2 -> potrebno je min 3 bita, 3\* 8 = 24 bita za niz*
      2. *q=5, slučaj floor(q) -> 0,1,2,3,4,0,1,1 -> potrebno min 3 bita, 3\*8 = 24 bita za niz*
   3. **Odredite restaurirane nizove za oba slučaja zaokruživanja?**
      1. *slučaj ceil(q) -> 0,5,10,20,25,5,5,10*
      2. *slučaj floor(q) -> 0,5,10,15,20,0,5,5*
   4. **Odredite kvantizacijski korak tako da duljina kvantiziranog niza bude 16 bitova (za oba slučaja zaokruživanja) ?   
      \***Napomena: max -> 11 (binarno) -> 3 (decimalno)



* + 1. *q = 8*

****

**20. Neka program sadrži dio O1 i dio O2 koji imaju udjele u vremenu izvođenja p1 = 35% i p2 = 65 %. Ukoliko se O1 može ubrzati 5 puta, a O2 3 puta, odredite koji dio se više isplati implementirati.**

****

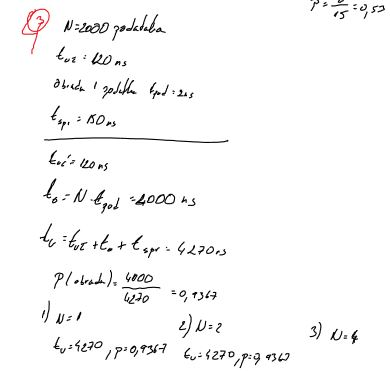
**Koliko će se ubrzanje dobiti uz optimizaciju oba dijela?**

Zna netko šta je N a šta U?1‌ ‌->‌   
// **N** -> broj puta koliko se može ubrzati izvođenje potprograma  
// **U** -> ubrzanje programa  
// **p** -> postotak, vrijeme izvođenja (dio programa **O** u omjeru **p** se može ubrzati **N** puta)

**U = 1/(p1/N1 + p2/N2) - računa se po toj formuli (barem piše na fer2)**

***//zapravo bi trebalo biti U = 1 / (1 - p1 - p2 + p1/N1 + p2/N2), al 1 - p1 - p2 = 0 pa ispadne //gornja formula***

**21. Algoritam obrade podataka sastoji se od početnog dijela koji učitava blok od 2000 podataka čije izvođenje traje 120 ns. Obrada jednog podatka traje 2 ns. Obrada podataka može se paralelizirati podatkovnim paralelizmom ako sustav ima više procesora. Nakon što su svi podaci obrađeni spremanje bloka obrađenih podataka traje 150 ns. Izračunajte ubrzanje i paralelnu efikasnost za sustave sa 1/2/4 procesora.**

****

**p = 0.9367 i t = 4270 za sva tri slučaja, samo se N mijenja (1/2/4)**

**//*zar ne bi trebalo biti t0 =4270, t1 = 2270 i t2 = 1270 (analogno za p-ove)? ja isto mislim***

***// ne bi trebalo biti, vrijeme ukupno se izračuna jednom i onda se izračunaLucia Raić***

***￼￼￼￼***

***18:18***

***a to je dosta lose dok ucim***

***18:18***

***onako potajno se nadam da ce sneaky snakes biti hit***

***18:18***

***i da cu moci od toga zivjeti***

***18:18***

***Lucia Raić***

***￼￼￼￼***

***18:18***

***a to je dosta lose dok ucim***

***18:18***

***onako potajno se nadam da ce sneaky snakes biti hit***

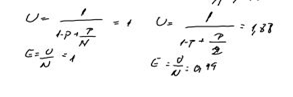
***18:18***

***i da cu moci od toga zivjeti***

***18:18***

***U i E za N = {1, 2 i 4}***

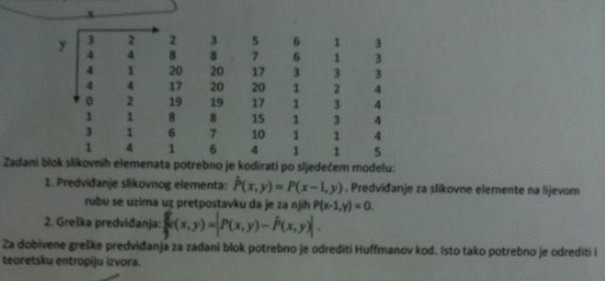
***// Vrijeme koje paraleliziraš paraleliziraš tako da ga uzmeš onakvo kakvo je u serijskom sustavu (1 procesor).***

****

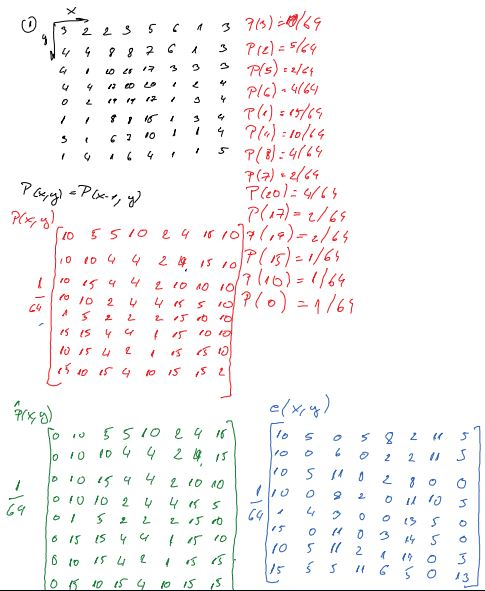
**U=1/(1-P+P/N)**

**E=U/N**

**21. Zadatak:**

****

Izračunati za svaki broj vjerojatnost pojavljivanja i napraviti matricu P(x,y) s tim vjerojatnostima. P^(x,y) je P(x,y) pomaknuta za jedan stupac u desno, tako da prvi stupac postaju nule. e(x,y) je apsolutna razlika P^(x,y) i P(x,y). I onda se za elemente e(x,y) računa vjerojatnost pojavljivanja i iz toga radi huffman.

****

Uzimaju li se ovdje vjerojatnosti pojavljivanja ili brojevi kao što su zadani na kraju?

* vjerojatnosti pojavljivanja, njima računaš entropiju
* ali u 30. nije tako :D
* cilj ti je iz tablice greške predviđanja konstruirati H. kod, jednom kada je dobiješ, prebroji koliko kojih elemenata ima unutra; dobiješ vjerojatnost pojavljivanja pojedinog elementa...iz toga konstruiraš H. kod i izračunaš njegovu entropiju (<http://denis-sofic.from.hr/huffmanov-kod>) i što se već traži u zad.
* Ma da, na to sam mislio. Za P(x,y) prepišemo elemente i onda tek na kraju tražimo vjerojatnosti, a ne odmah na početku
* Točno
* (PALAC GORE)
* xD

**Ili ja nešto krivo brojim ili je P(4) =11/64, P(2) = 4/64, P(17)=3/64. (malo su krivo izbrojani neki brojevi) - tenks mejt na potvrdi :)**

**tipa za objašnjenje**

***ovo je krivo riješeno, scroll dolje na zadatak istog(30. )***

**Jel zna netko kako sad Huffmana dobiti? Jel to onim stablom ili entropijskim**

**kodiranjem(DC,AC)?  
- mislim da ide stablo, barem je tako rj na materijalima**

**22. Zadatak:**

**Objasnite značaj aritmetičkih naredaba koje imaju izvedenu aritmetiku sa zasićenjem te navedite gdje se takve naredbe mogu efikasno koristiti.**

(MAS-JK-3-4, slajd 11, 12 i 13) slide 16! -> slide 23!

<https://en.wikipedia.org/wiki/Saturation_arithmetic#Modern_use>

Vidljivo je da two's complement sumacijom dva velika pozitivna broja dobivamo negativni broj, što predstavlja značajnu pogrešku. Ovakva pojava je vrlo neugodna jer uzrokuje oscilacije gotovo maksimalnom amplitudom iako je na ulazu nula. Ona se može spriječiti tako da se umjesto klasične two's complement aritmetike koristi aritmetika sa zasićenjem (saturation). Njenom primjenom, suma dva velika pozitivna broja koja bi u normalnom slučaju rezultirala negativnim brojem, zamjenjuje se s maksimalnim pozitivnim brojem. Analogno, suma dva velika negativna broja, zamjenjuje se s maksimalnim negativnim brojem. (preuzeto iz jednog PDF-a)

Obični procesori mogu izvesti jednu ALU operaciju (širine koju ima ALU) po periodu. Kako bi ubrzali izvođenje moramo mnogo pažnje posvetiti dohvatu podataka te optimizacijama petlji.

Korisno za npr. DCT ako procesor ima sklopovski izvedeno množenje i pripadnu naredbu (npr značajna prednost ako postoji naredba množenja sa zbrajanjem – MLA – osnovna karika kod leptir operacija prilikom računanja npr. DCT, DFT)

Efikasno korištenje protočne strukture: Loop-unrolling, Branch prediction…

**23 Zadatak:**

**Koji su osnovni nedostaci asimetričnih algoritama za kriptiranje? Kako se ti nedostaci rješavaju u stvarnim primjenama?**

Različiti ključevi za kriptiranje i dekriptiranje. (MAS4\_2016, 39. slajd)

Ne znam koji su nedostaci i kako se rješavaju.

i

Najveći problem je vjerojatno brzina. Asimetrični algoritmi s dva ključa dosta su sporiji od simetričnih sa secret keyem. Vjerojatno je improvement izvedba kriptografske jedinice na HW razini pa se klijentski CPU ne mora zamarati s dekripcijom i enkripcijom. [odavde <http://www.fer2.net/showpost.php?p=2370374&postcount=25>]

Osnovni nedostaci asimetričnih algoritama su problemi razmjene javnih ključeva, potvrde identiteta (da li je taj javni ključ uistinu od osobe A) te kako kriptirati jako dugačke poruke. Prvi i drugi problem se rješava uvođenjem CA (Certificate Authority) koji potvrđuje i jamči autentičnost certifikata u kojem se nalazi javni ključ. Treći problem se rješava kriptiranjem poruke simetričnim algoritmom čiji se ključ zatim kriptira asimetričnim algoritmom i pošalje drugoj strani.

**24 Zadatak:**

**Koji su osnovni problemi koji su uzrokovali zastoj u povećanju performansi jednojezgrenih računalnih arhitektura?**

Pregrijavanje, manje cache memorije dostupno ili je potrebna jako velika memorija što zauzima puno prostora na dieu, usporen rast i efikasnost. [odavde: <http://www.fer2.net/showpost.php?p=2370374&postcount=25>]   
<http://superuser.com/questions/152011/why-multi-core-processors>

**25. Zadatak**

**Objasnite svojstva i međusobne razlike između medijskog koprocesora i medijskog procesora?**

(MAS4\_2016)

**Medijski procesor**: procesor se projektira PRVENSTVENO za obradu multimedijskih podataka, a ostale stvari može obrađivati ali tek sporedno. Projektira se s ciljem visokih performansi za određen skup algoritama, to je još uvijek PROCESOR: još uvijek je programibilan. Prednosti: promjena algoritama, dodavanje funkcionalnosti,....

Izuzetno visoke performanse (veće od GPP, medijskih koprocesora). Složen postupak programiranja. Paralelno izvođenje operacija unutar jedne naredbe. Podrška za opće zadatke mora biti osigurana od dodatnog procesora.

**Medijski koprocesor**: posebni procesor koji se stavlja u sustav da bi se u njemu obrađivali zahtjevni dijelovi algoritma.

U osnovi : DSP sa mnoštvom periferija. Jeftiniji od procesora opće namjene. Performanse prilagođene aplikaciji (manje od GPP). Manja potrošnja. Programabilan !! (mogućnost poboljšanja i dodavanja aplikacija). Predviđen za porodicu algoritama. 32-bitovni CPU (npr. ARM) obično dobar za ostale poslove (mreža, user i/f, ...)

**i**

**26. Zadatak:**

**Koja je optimalna širina pojedinog podatka kod SIMD naredaba za obradu video podataka?**

(MAS-JK-3-4, slajd 15)

8 bita video, 16/32 audio

**27. Zadatak:**

**Objasnite osnovna svojstva i razliku između sljedeća 2 pojma: deadlock i livelock.**

(MAS-JK-3-4, slajdovi 7 i 8)

Deadlock refers to a situation when a thread waits for an event that never occurs. This is usually the result of two threads requiring access to resources held by the other thread.

Livelock refers to a situation when threads are not making progress on assigned computations, but are not idle waiting for an event.

**28. Zadatak:**

**Usporedite korake te prednosti i nedostatke u JPEG i Wavelet kompresiji.**

JPEG: RGB u YUV, pomak, 2DCT, kvantizacija

d   
ovdje ima nešto o waveletu: Kod wavelet transformacije, cijela se slika komprimira, tj. ne dijeli se na blokove kao što je slučaj kod diskretne kosinusne transformacije.

JPEG tj DCT transformacija je bolja za niske frekvencije dok je wavelet kompresija bolja za uzorke koji se rijetko pojavljuju. Wavelet se koristi kod JPEG2000.

Wavelet transformacija daje bolje rezultate za ljudsko oko – osigurava veću

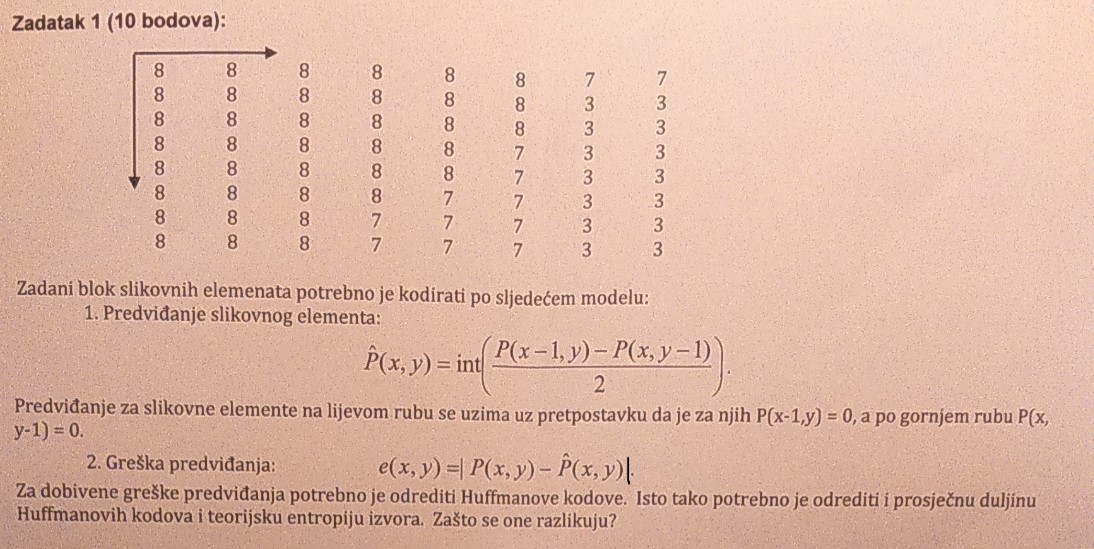
vremensku razlučivost za visoke frekvencije, a visoku frekvencijsku razlučivost za niske frekvencije.

Omogućava i progresivno kodiranje. To znači da se svim koeficijentima slike prvo kodiraju najviši bitovi pa se onda postupno kodiraju niži bitovi čime se povećava kvaliteta rekonstruirane slike. Upravo to omogućava i skalabilnost, što je vrlo korisno jer se u bilo kojem trenutku može prekinuti s dekodiranjem, a ipak će se rekonstruirati cijela slika (doduše, ne cijela s jednakom kvalitetom, ali i to ponekad bude dovoljno za određene primjene). Važna prednost je i multirezolucija, tj. moguće je rekonstruirati sliku za razne vrijednosti rezolucije manje od originalne. (preuzeto iz jednog PDF-a)

**29. Zadatak:**

**Zašto se korak kvantizacije obavlja nakon DCT, a ne prije DCT?**Zato što se ovime može različito kvantizirati frekvencije, tj. kad bi kvantizaciju izveo prije DCT ne bi znao što je što na slici i ako bi išta dijelio promijenilo bi sliku neprepoznatljivosti. Ako je nakon DCT-a obradiš možeš reći koje frekvencije ćeš isfiltrirati da je slika i dalje razumljiva i prepoznatljiva.

**30 Zadatak:**

￼￼

Ove je 12 v prvem stopcu, sam se male zajebel - jel ovo slovenski? (prijevod: Ovo je 12 u prvom stupcu, malo sam pogrijesio) [prigorski]

u ovom rjesenju se nisu prvo računale vjerojatnosti kao u istom ovakvom ali par stranica prije, znaci P(8) = 38/64, P(7)=12/64, nez kaj je tocno al me zanima zas je to tak, mozda zato ne štima?

Dakle, ovaj P je vrijednost u zadanom bloku, ne nekakva vjerojatnost. Ovaj zadatak je model predviđanja slike u videu (prezentacija Kompresija videa). Računa se predviđena vrijednost novog bloka na osnovu neke funkcije i gleda kolika je greška u odnosu na stvarnu vrijednost. Kodira se greška jer greška ima puno manje različitih redundantnih informacija za kodirati (gle prezentaciju). Zadatak iznad di se prvo računala vjerojatnost pojavljivanja je neispravno riješen. Drugim riječima, ovaj (zad 30) je dobro riješen...osim ovog čuda sa proizvoljnim Huffmanom. :)

// Err, ovaj Huffman je čudan. Malo ide gore 1, malo ide gore 0, wat? (slažem se da tu nešt ne valja, treba biti ili konstantno 1 kod većeg čvora ili konstantno 0, ne može biti malo 1 malo 0; ostalo je sve ok)

// Bas sam pogledao u biljeznici iz TINF-a, znaci 1 ide na onu granu gdje je veca suma, a 0 na onu koja je manja suma. Dobro rjesenje bi bilo ovo:

8: 0

7: 110￼

3: 101

12: 100

1: 1110

4: 11111

5: 11110

Moze netko potvrditi?  
  
POTVRDA! Tako smo uvijek do sad radili :)

<http://denis-sofic.from.hr/huffmanov-kod> - objasnjenje entropije i prosjecne duljine kodne rijeci

**31 Zadatak:**

**Na sljedećem primjeru objasnite i izračunajte 16-bitnu aritmetiku sa zasićenjem:  
 02F5 5478 (16)  
+F8DC BAA0 (16)**

Mislim da se tu ne smije desiti overflow nego ako predje FFFF da se stavi FFFF ko vrijednost i s obzirom da se radi o 16 bitnoj aritmetici da se zbroje po 16 bitova odvojeno.  
**Ok i kako bi rješenje ovog izgledalo???**

**FAE1FFFF? ako netko moze potvrdit -> nije to dobro**

**Točan rezultat je: FBD1FFFF, radi se o 16bitnoj aritmetici.**

**0**

**Sto nije da zbrajamo 16 po 16 bitova? Dalke FDB2FFFF (kada zbrojimo najdonjih 16 bitova, predje se tih 16 pa se napisu sve jedinice tamo, tj. bude FFFF, a carry se prenosi dalje, zar ne? Onda zbrojimo 02F5 i F8DC i dobijemo FBD1, plus onaj carry je FBD2. Jel ovo tocno?)**

**Mislim da se carry ne prenosi jer svakih 16 bita je svoja operacija.**

**Jel moze netko objasnit kako se ovo zbraja jer ja dobivam FBD20F18 kad zbrajam “normalno”..**

**Koliko sam skužio, prvo zbrojiš donjih 16 bita koji daju 5478 + BAA0 = 10F18, kako to overflowa - zaokružiš na FFFF. isto tako i za drugih 2. F8DC + 02F5 = FBD1. RJ.: FBD1 FFFF**

**iz multeha (**[**http://www.fer2.net/showpost.php?p=1796573&postcount=19**](http://www.fer2.net/showpost.php?p=1796573&postcount=19) **- tema multimedijske tehnologije, znam da sam dobio sve bodove tu..) : sa zasićenjem :  
  
1 2345 6789+  
 FEDC 0A98   
=  
1 FFFF 7221**

**a s wrap aroundom:  
  
1 2221 7221**

**//znaci carry se ne prenosi? prema ovom iz multeha ne. ali to je nečji post s fer2 isto.**

**Da, carry se ne prenosi. Knezovićeva prezentacija br.4 objašnjava da to što zbrajamo nisu pojedinačni brojevi od 32 bita, nego više pojedinačnih brojeva od 16 bita upakiranih u 32 bita radi bržeg izvođenja (zbrajamo 2 16-bitne vrijednosti u jednom ciklusu umjesto u 2).**

### ^^Prezentacija MAS-JK-3-4 str.23

**32. Zadatak:**

**Za općeniti procesor napisati potprogram** *int ReadBlockFromSCCD (unsigned char device\_adr, unsigned char reg\_subadr, unsigned int count, unsigned char \*buffer)***. Navedena funkcija čita niz uzastopnih unutarnjih registara uređaja spojenog na SCCD sabirnicu. Parametri funkcije su: device\_adr - adresa uređaja na sabirnici, reg\_subadr - adresa internog registra koji se čita, count - broj uzastopnih registara koji se čita, buffer - pokazivač na polje pročitanih podataka. Funkcija vraća broj pročitanih podataka ili -1 u slučaju greške. Na općem procesoru postoje funkcije za čitanje i pisanje na I2C sabirnici:** *int I2CRead(unsigned char I2C\_adr, unsigned int count, unsigned char \*buffer)* **i** *int I2CWrite (unsigned char I2C\_adr, unsigned int count, unsigned char \*buffer)***. Parametri funkcija su: I2C\_adr - adresa I2C uređaja, count - koliko se podataka čita ili piše, buffer - pokazivač na polje pročitanih podataka ili na polje podataka koje treba zapisati. Funkcije vraćaju broj poslanih/primljenih podataka ili -1 u slučaju greške.**

Dakle, kolko sam ja shvatio taj I2C i obzirom kako smo napisali u labosu, ta fja bi trebala izgledat ovako:

int ReadBlockFromSCCD (unsigned char device\_adr, unsigned char reg\_subadr, unsigned int count, unsigned char \*buffer)

{

int i;

for (i = 0; i < count; i++)

{

if (I2CWrite(device\_adr >> 1, 1, &reg\_subadr) == -1)

return -1;

if (I2CRead(device\_adr >> 1, 1, buffer+i) == -1)

return -1;

reg\_subadr++;

}

return i;

}

Uređaju se prvo treba poslati adresa registra kojeg želimo čitati, zatim pročitamo taj registar. Jedino što ova funkcija pretpostavlja da su svi registri 8-bitni. Kad ne bi bili, morao bi se i povećavati za kolko god vrati ovaj I2CRead umjesto za 1. Ovaj device\_adr treba shiftati udesno za 1 jer I2C koristi 7-bitne adrese, a najniži bit je samo jel pišemo ili čitamo, što bi trebalo biti interno riješeno ako imamo odvojene funkcije za read i write. Bilo bi dobro na ispitu napisati napomenu za ovo shiftanje da vam ne skinu bodove na tome.

// Malo mi čudno se čini ovako kratak kod za 10 bodova :P

// A gle, uzmi u obzir što bi napisao netko tko nije radio na tom dijelu labosa

// Može i kraće:

*int ReadBlockFromSCCD (unsigned char device\_adr, unsigned char reg\_subadr, unsigned int count, unsigned char \*buffer)*

*{*

*if (I2CWrite(device\_adr >> 1, 1, &reg\_subadr) == -1) return -1;*

*return I2CRead(device\_adr >> 1, count, buffer);*

*}*

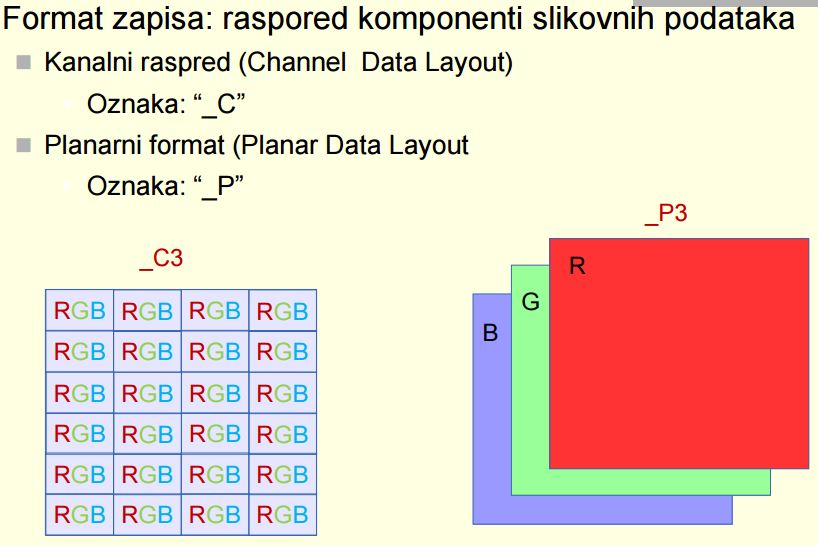
// Znamo da se ovako može pisati na SCCB, ali jel sigurno važi i za čitanje?

//koliko sam shvatio iz prez, sa čitanjem ne možeš adresirati pojedini registar unutar adrese nego se čita onaj registar koji je bio zadnji adresiran sa write-om. tako da po meni je ovo ok

// Ma ne to, nego kad pišeš I2Com na SCCB, ako pišeš više bajtova od veličine registra, svi slijedeći bajtovi idu u uzastopne registre (piše na prezentaciji nekoj), ali ne piše da se ista stvar može sa čitanjem (čitati više bajtova od veličine registra pa dobiješ nazad sve u sljedećem uzastopnim registrima)

**33 Zadatak:**

Objasniti i skicirati razliku između pikselnog i planarnog rasporeda u biblioteci IPP. [MAS-JK-3-1]



// može netko objasniti o cem se tu radi?

Znači, planarni raspored podataka je da prvo imaš cijelo polje crvenih vrijednosti, pa cijelo polje zelenih vrijednosti, pa cijelo polje plavih. Kanalni raspored je standardno, jedno polje sa RGB vrijednostima. Treba još to malo po prezentaciji proći i probat naći…a ako ne u prezentaciji, onda u dokumentaciji IPPa.

In channel format, all values share the same buffer and all values for the same pixel position are interleaved together. For planar format, there is one value per pixel but potentially several related planes.

Planarno:

// RRRRRRRR  
// RRRRRRRR  
// RRRRRRRR

// RRRRRRRR  
// GGGGGGGG  
// GGGGGGGG  
// GGGGGGGG  
// GGGGGGGG  
// BBBBBBBB  
// BBBBBBBB  
// BBBBBBBB  
// BBBBBBBB

Pixelno(kanalno):

// RGBRGBRGBRGBRGBRGBRGBRGB

// RGBRGBRGBRGBRGBRGBRGBRGB

// RGBRGBRGBRGBRGBRGBRGBRGB

// RGBRGBRGBRGBRGBRGBRGBRGB

**ZI 16/17**

34. Trebalo je dohvacati sliku s kamere koja vraca sliku u formatu **RGB 5:6:5**, a trebalo je koristiti funkcije **CAMERA\_VSYNC()** koja vraca 1 ako je VSync high, inace 0, **CAMERA\_HREF()** vraca 1 ako je HREF 1, slicno i za **CAMERA\_CLOCK()**. Dakle trebalo je primiti njene podatke i to spremiti u normalnom RGB formatu u neki niz.

**sto je RGB 5:6:5? 5 bitova za crvenu, 6 bitova za zelenu i 5 bitova za plavu, tnx**

(kod odozdo u fileu, citljivije)

<https://www.fer.unizg.hr/_download/repository/dohvat_slike.c>

while(CAMERA\_isVSYNdown(READ\_CAM));

while(CAMERA\_isVSYNup(READ\_CAMe)) {

for(y = 0; y<480; y++){ //288

//Wait line to start

do{

tt=READ\_CAM;

if(CAMERA\_isVSYNup(tt)){

printf("VSINC error [%d]\n",y);

goto ll;

}

}while(CAMERA\_isHREFdown(tt));

//Y[z] =(u8)(tt);

//z++;

for(r = 0;r<1280;r++){ //352

do{tt=READ\_CAM;}

while(CAMERA\_isPCLKdown(tt)); //while(CAMERA\_isPCLKup(tt));

//---------

do{

tt1=READ\_CAM;

if(CAMERA\_isHREFdown(tt1) && r!=1279){

printf("HREF error [%d]\n",r); goto ll;}

}

while(CAMERA\_isPCLKup(tt1));

//Write data

//HMY[z] =(Xuint8)(tt>>3);

Y[z] =(u8)(tt);

z++;

}

while(CAMERA\_isHREFup(READ\_CAM));

}

//xil\_printf("%d",z);

printf("OK !!!\n");

35.Bila je zadana deklaracija funkcije

**convert(unsigned char \*source, unsigned char \*\*destination, int width, int height)**.

Dakle, ova funkcija prima pokazivac na jednodimenzionalan niz piksela u formatu kanalni 24-bitni RGB (**source**), sirinu i visinu slike u pikselima (**width**, **height**) i pokazivac na niz pokazivaca koji moraju sadrzavati rezultat konverzije (**destination**).

Unutar funkcije trebalo je obaviti pretvorbu iz RGB kanalnog u YCbCr planarni koristeci one IPP funkcije. Ponudili su nekoliko funkcija i naveli njihove parametre pa je trebalo tocno odabrati funkciju koja nam je potrebna za ovu konverziju, a to je funkcija **ippiRGBToYCbCr\_8u\_C3P3R**.

convert(unsigned char \*source, unsigned char \*\*destination, int width, int height){

Ipp8u \*ipp\_source = (Ipp8u \*) source;

Ipp8u ipp\_destination[3][width\*height] = {0};

IppiSize roiSize = {width, height};

int srcStep = width\*3;

int dstStep = width\*3;

IppStatus st = ippStsNoErr;

st = ippiRGBToYUV\_8u\_C3P3R(ipp\_source,srcStep,ipp\_destination,dstStep,roiSize);

destination = ipp\_destionation;

}

**1) Zašto se korak kvantizacije obavlja nakon DCT, a ne prije?**

1) Zato što se ovime može različito kvantizirati frekvencije. Tj. kad bi kvantizaciju izveo prije DCT ne bi znao kaj je kaj na slici i ako bi išta dijelio promijenio bi sliku do neraspoznatljivosti. Ako je nakon DCT-a obradiš možeš reći koje frekvencije ćeš isfiltrirati da je slika i dalje prepoznatljiva.

**2) Objasnite značaj aritmetičkih naredbi koje imaju izvedenu aritmetiku sa zasicenjem, te navedite gdje se takve naredbe mogu efikasno koristiti?**

JK 3-4, mogu se efikasno koristiti kod SIMD modela za zbrajanje vise podataka u jednom ciklusu...

**3) Koji su osnovni nedostaci asimetričnih algoritama za kriptiranje? Kako se ti nedostaci rješavaju u stvarnim primjenama?**

3) Najveći problem je vjerojatno brzina. Asimetrični algoritmi s dva ključa dosta su sporiji od simetričnih sa secret keyem. Vjerojatno je improvement izvedba kriptografske jedinice na HW razini pa se klijentski CPU ne mora zamarati s dekripcijom i enkripcijom.

**4) Koji su osnovni problemi koji su uzrokovali zastoj u povećanju performansi jednojezgrenih racunalnih arhitektura?**

4) Pregrijavanje, manje cache memorije dostupno ili je potrebna jako velika memorija što zauzima puno prostora na [die-u](https://en.wikipedia.org/wiki/Die_(integrated_circuit)), usporen rast i efikasnost.

**5) Objasnite svojstva i međusobne razlike između medijskog koprocesora i medijskog procesora?**

5) Nisam siguran. Rekao bih da je koprocesor u sustavu s general purpose procesorom i pomaže mu obraditi MM zadaće. MM coproc ima veći broj stream procesora dok glavni proc ima manji broj jezgara i veću složenost. Primjer bi vjerojatno bili prva računala koja bilo moguće imati u kući poput Amige.

Prema toj logici medijski procesor bi vjerojatno bila grafička kartica koja je direktno projektirana tako da može obrađivati multimediju.

**6) Koja je optimalna širina pojedinog podatka kod SIMD naredaba za obradu video podataka?**

6) Nisam siguran. Bubnuo bih da bi to trebala biti veličina L1 cachea kako bi procesor minimalno čitao iz L2 ili dalje. Ispravite me...

**7) Objasnite osnovna svojstva i razliku izmedju deadlock-a i livelock-a?**

7) Livelock je kada dva procesa rade i mijenjaju, ali ne mogu se odlockati dok se kod deadlocka oba statično vrte. RL primjer bi bio 'canadian standoff' tj. dvije osobe pokušavaju proći kroz vrata, ali puštaju jedno drugome prednost pa onda probaju u isto vrijeme itd. You get the point. A deadlock je kad dva žaka u isto vrijeme prođu kroz vrata pa se zaglave i niti jedan nemre proći.

**8) Usporedite prednosti i nedostatke u JPEG i Wavelet kompresiji?**

8) JPEG tj DCT transformacija je bolja za niske frekvencije dok je wavelet kompresija bolja za uzorke koji se rijetko pojavljuju. Wavelet se koristi kod JPEG2000.

