0 7

- *]/2009-10/*
-

Dokument za pripreu MI i ZI iz Multimedijske arhitekture i SUS (☺)tava-a

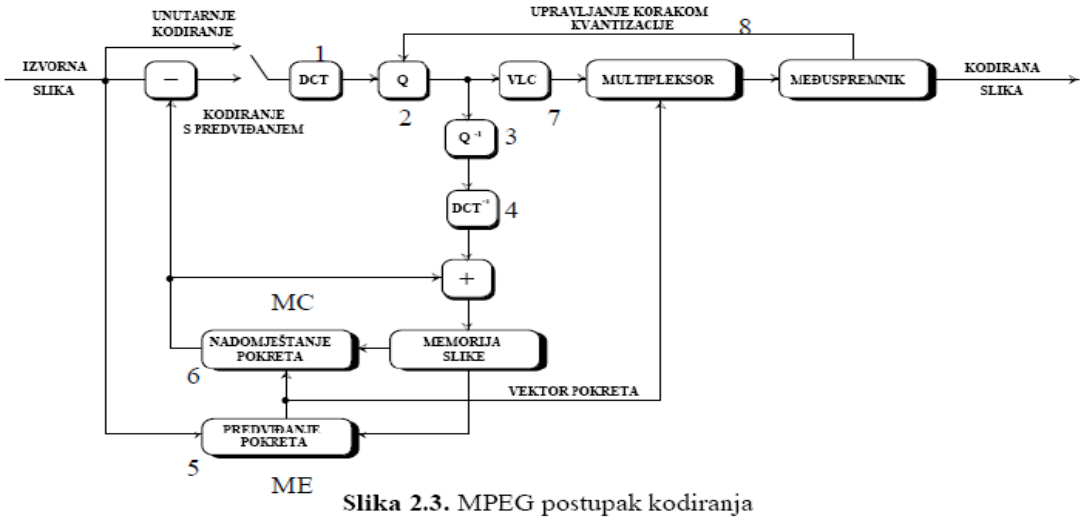
NAPOMENA: zadatke s programiranjem ne treba znati - Čak ni za završni?? naaah

- ***ali treba znati IPP funkcije, rekao Alen Duspara na MSTeamsima - ima li toga u prezentacijama?***

TODO: Pronaći na materijalima koji su Šablonski zadaci koji se često ponavljaju na ZI iz MAS-a

Zadatci za MI

1. **MI: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 kvantizacije**. (Na slici to je ova osmica) => u kojoj prezentaciji je ova slika? => ima slična slika 44. slajd treće prezentacije prvog ciklus
   -



Slika 2.3. MPEG postupak kodiranja

2. **MI: 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

3. **MI: Za128 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) + ;

- Za svaki piksel se mora obaviti 9 množenja i 9 zbrajanja. 1024*768 = 786432 piksela
- Dakle 1024*768*9 zbrajanja i 1024*768*9 množenja.

4. **MI: 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).**

Ne bi li ovdje trebalo biti (4*2 + 1) (umjesto 8*2 + 1) kod 8x8 bloka?
Mislim da je oke jer ako imas 8x8 blok i možeš ić još 8 lijevo i 8 desno onda je područje 24x24 a blok 8x8 se može 17 puta prošetati po retku i po stupcu (24-8+1=17) pa je to tih 17*17 tj (2*8+1)*(2*8+1)
To onda znači da gledamo i nule izvan bloka? Ne znam koliko to ima smisla, za 16x16 mi ima smisla da možemo 17 po retcima i stupcima (logika sa slajda 17 iz prezentacije MAS-3-2)
Nez mislim da nema nikakvih nula al da logika je ista za 16x16 di ima 16+8+8=32 prozor pretr pa ima (32-16+1)(32-16+1) = 17*17 pretrazivanja

Za blok 8×8 računamo MAD:
8 * 8 = 64 oduzimanja (gledamo kao zbrajanje)
1 množenje
8 * 8 - 1 = 63 zbrajanja

Sad koliko puta ćemo puta napraviti MAD **(ovisi o pomaku)**
(8 * 2+1) * (8 * 2 + 1) * (64 + **63** + dohvat) = 36703 zbrajanja + dohvati
(8 * 2 + 1) * (8 * 2 +1) * (1 + dohvat) = 289 mnozenja

Npr da imamo 16×16 blok onda bi :
16 * 16 = 256 oduzimanja (gledamo kao zbrajanje)
1 množenje
16 * 16 - 1 = 255 zbrajanja

I pod uvjetom da je pomak +-8
(8 * 2+1) * (8 * 2 + 1) * (256 + 255 + dohvat) = 147679 zbrajanja + dohvati
(8 * 2 + 1) * (8 * 2 +1) * (1 + dohvat) = 289 mnozenja

5. **MI: Navesti barem tri algoritma 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)
   - One at a Time (OTA) //ne znam spominje li se u .ppt

6. **MI: 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)

7. **MI: Koji su nedostaci brzih algoritama za procjenu pokreta?**
   - Ne daju uvijek najbolji rezultat, jedino kod potpunog pretraživanja smo sigurni da je stvarno nađen najsličniji blok

- Iz preze: veće razlike koje se trebaju kodirati, a time je i izlazna količina podataka veća (manji stupanj kompresije) - 3. preza, 59. strana

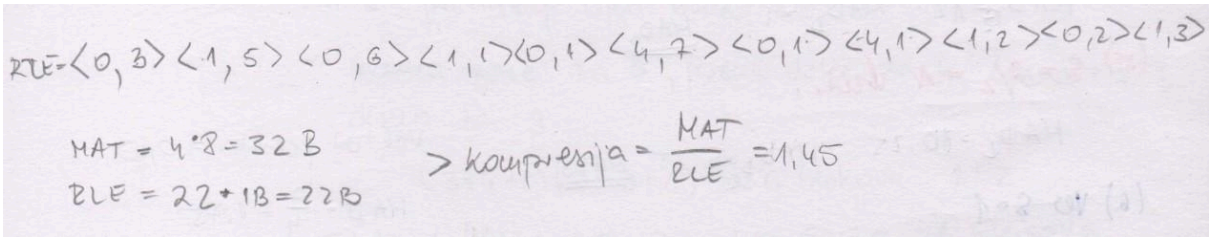
8. **MI: DCT transformacija: Zadani su prototipovi dviju funkcija. Potrebno je implementirati obje funkcije u programskom jeziku C (kompletni program). Prva funkcija kao argumente prima pokazivač na niz podataka *data, veličinu polja podataka N i M te pokazivač na kvantizacijsku tablicu *qtable veličine 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.**

//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]

9. **MI: Zadan je blok podataka:**

0	0	0	1	1	1	1	1
0	0	0	0	0	0	1	0
4	4	4	4	4	4	4	0
4	1	1	0	0	1	1	1

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.



   a. ***Što predstavlja MAT i kako se dobije ovih 4*8?***
      i. MAT je zadana tablica s brojevima koja ima 4 reda i 8 stupaca, a svaki broj je jedan B (byte)
   b. ***otkud 22*1B?***
      i. kod RLE <S,N>, svaki S je 1B i svaki N je 1B, u ovom slučaju je 11 S i 11 N što nam daje ukupno 22B
   c. ***Kad bi npr. nule bile na kraju retka i na početku sljedećeg jel ih sve pišemo u jedne zagrade <,>?***
      i. Sve se pise u jedne zagrade, jer u koder ulazi kao stream
      ii. "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"
   d. Kako bi bili sigurni vjerojatno možete komentirati da u slučaju da je bitno dobiti blok natrag onda je omjer kompresije malo manji jer nekako treba zapisati dimenzije bloka, ali koliko manji je implementacijski detalj i nema veze sa specifikacijom RLE

10. **MI: 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 ?
    - Teški je bullshit da se K koristi za 1024 - mi koristimo SI standard koji nedvosmisleno kaže da se za kilo koristi prefiks k, a za kibi prefiks Ki, o čemu govori ISO 80000. Tako da je **točno rješenje** $1920 \cdot 1080 \cdot 24\,s^{-1} \cdot 3B = 149299200\,B/s = 149.3\,MB/s$, tj. omjer kompresije 29.86. Da to nije slučaj trebalo bi biti jasno naznačeno u zadatku jer bi se koristilo nešto što nije međunarodni dogovor koji poštujemo, ali i praksa koja je tu već nekoliko desetljeća.
    - **kilo je 1000**, ako ti skinu bodove pravac uvidi i srat oko toga. Ako pretpostaviš da kilo = 1024, tad se pozivaš na javnu percepciju, ako pretpostaviš da kibi = 1024 tad se pozivaš na konkretne norme, o čijoj se važnosti priča još u prvom predavanju
    - Zna netko odgovor na drugo pitanje, vezano za stupanj kompresije? 29.86 je odgovor
    - Opće je poznato da se brzine prijenosa pretvaraju s 1000 dok se memorijske jedinice pretvaraju s 1024 - bilo je zadano u zadatku

11. 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…

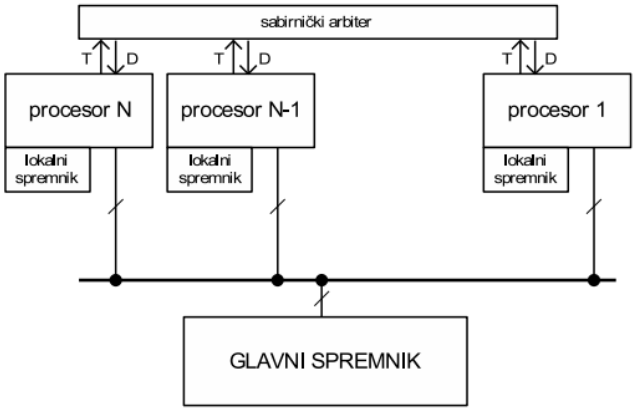
// koje nije odgovor? ovo ispod?

Kod **višeprocesorskih** sustava, svaki procesor **ima svoju** cache memoriju i sabirnicu. //**ovo je krivo :)**

Odgovor pod a) je točan Osnovni koncepti operacijskih sustava SLAJD 28 pa nadalje

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

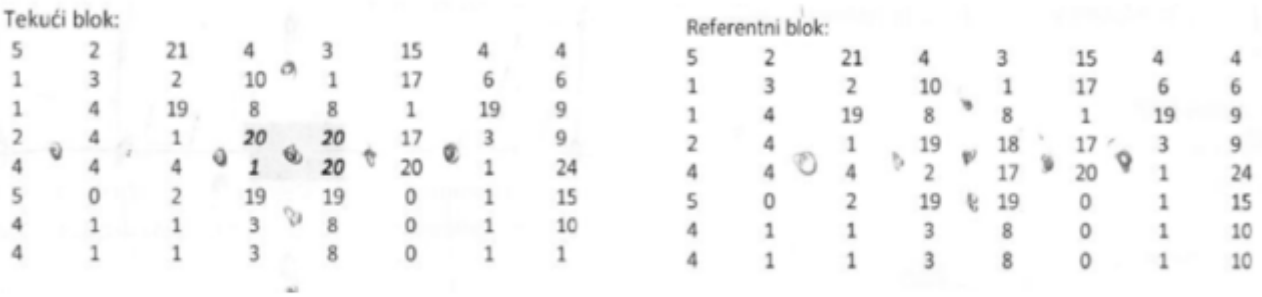
-> netocno; ovisi o arhitekturi sustava:



Slika 3.7. Višeprocesorski sustav

- 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).

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



(2020/21) DA SKRATIM MUKU BUDUĆIM GENERACIJAMA:
(Nakon rasprava na forumu i rezultata ZI-a)

DISTANCE

Distance d je MAX udaljenost od ruba bloka do zadnjeg stupca/retka. Znači ako je blok u "sredini", a matrica 8x8 onda d=3. Ako je matrica 10x10 d=4. Ako je 11x11, s jedne strane ce udaljenost bit 4, s druge 5, ali se uzima max, znaci d=5.

STEP
Za ORT i LOG pretraživanja je početni step s=d/2. Pritom se zaokružuje na manju vrijednost, ako je d=3 onda je početni s=1.
Za 3SS search se početni step računa prema formuli s = 2^(N-1). Pošto je THREE step search onda je N=3 i onda s=4.

ALGORITMI
Za vježbu preporučujem pogledat ove dvije slike gdje su ručno riješeni ort i log pretraživanja s bojama. NAPOMENA da je na tim postupcima d pogrešan, treba bit 4, pa početni step 2, ali za vježbu se pravite da je točno jer ostatak postupka s micanjem po blokovima štima.
Za same postupke algoritma gledajte screenshotove što su uzeti s prezentacija predmeta. To i praćenje postupka s ovog ručno riješenog primjera će bit dovoljno za razumijevanje, ali evo ukratko razlika između algoritama:

ORT:

1. **Izračunaj grešku za trenutni blok. Step=d/2**
2. **Izračunaj grešku za horizontalne blokove udaljene za step od trenutnog bloka. Nađi minimalnu grešku. Ako je greška nekog od ovih blokova manja od greške od trenutnog, prebaci se u taj blok i od njega nastavi dalje. Ako ne, ostaje se u istom bloku**
3. **Izračunaj grešku za vertikalne blokove udaljene za step od trenutnog bloka. Opet nađi min i prebaci se ako treba**
4. **Step = step/2 Ponavljaj korake 2 do 4 dok step ne bude 1 (i za njega se računa)**

LOG:

1. **Izračunaj grešku za trenutni blok. Step=d/2**
2. **Izračunaj grešku i za horizontalne i za vertikalne blokove udaljene za step od trenutnog bloka. Tek onda nađi minimalnu grešku, prebaci se ako treba**
3. **Ako se prebacilo u novi blok, vrati se na korak 2**
4. **Ako se ostalo unutar istog bloka step=step/2**
5. **Ponavljaj korake 2-4 dok s>1**
6. **Kad step postane 1 računa se greška na 8 lokacija oko trenutnog bloka, pogledajte sliku ručno urađenog LOG zadatka za ovaj zadnji korak za dobru vizualizaciju**

3SS:

Dovoljno je samo pogledat sliku s preze, imaju 3 koraka, početni step je 4 i polovi se u svakom koraku. Nakon izračuna početne greške, za svaki korak će se računat greške na 8 lokacija oko trenutnog bloka udaljene za step.

U zadacima se još može pitati broj računanja. Samo se izbroje svi blokovi za koje se tražila greška, s tim da se ne broji ako se ponovio isti blok.

I to je to. Dolje je sve diskusija oko t e čuvene vrijednosti za d, ostavit ću je jer je smiješno kako s dužim scrollanjem postaje sve kaotičnije i šarenije jer su morali zapaprit ove godine ispit na način da bude na zaokruživanje i da ti odu bodovi na pola zadataka ako pogriješiš prije nego što i počneš provodit algoritam. U svakom slučaju definitivno ovaj dio gradiva treba znat jer su ove godine skoro pola bodova nosili. Dovoljno je jednom s razumijevanjem preć jedan zadatak a za ostale znat pseudokod. Sretno!

ALGORITAM JE TU DVIJE STRANICE ISPOD, SAMO TREBAŠ PRATITI ALGORITAM

Pretpostavka da gornja 2 bloka podataka predstavljaju ulaza okvira 2x2 i ako pretpostavimo algoritam pretrazivanja 3SS (Three Step Search), izračunati vektor pomaka za označeni blok u g ornjem okviru. Koristiti MAD kao mjeru poremećaja.

VEKTOR POMAKA (0,0) JER JE BLOK OSTAO ISTI - blok se ne miče jer je MAD tog istog bloka manji od svih ostalih*

- *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 (*)*

***Full search / "*pri cemu je velicina prozora 6 slikovnih elemenata"**

- *6 elemenata znači da MAD = ⅛ * (|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.*

Može netko objasniti razliku između ORT/LOG/3SS u nekoliko riječi? :)

ORT: gledaš lijevo i desno i sredinu, potom od minimuma gore i dolje, nakon tog se smanjuje korak AKO SAM JA DOBRO SHVATILA, trebalo bi onda ponavljat to dok ne dođeš do min, dakle ide horizontalno pa vertikalno pa smanjivanje koraka i repeat

LOG: uvijek gledaš lijevo-desno/gore-dolje, ako je min u sredini smanjuješ korak, u zadnjem koraku isto sve oko najmanjeg gledaš

3SS: 3 su koraka, u prvom gledaš taj srednji blok i n, ovisno o udaljenosti, 8 oko njega + 1 najegovom mjestu (slika nacrtanog

algoritma!) pa ideš korak bliže tom min izračunatom bloku (postane novi centar od 9 blokova kao u prvom koraku) opet računaš 8 oko novog centra ako ih već nemas izracunate, i onda isto radis i za treci korak. Smanjuje se koliko se daleko gleda od centralnog bloka od tih 9 svakim korakom na pola prema pseudokodu/algoritmu

Kako znam koji smjer treba odabrati u svakom od njih? -ides prema minimumu nakon svakog koraka, dobro objasnjeno u onom pdfu sa rjesenim meduispitom iz 15/16
-ima na videu treceg predavanja na 1:04:17 kako je objasnio, poslje toga pogledaj pseudokodove na kraju dokumenta

Molim vas, pomozite mi, zbog čega je step size u početku postavljen na 3?
x
u algo pise da je korak 2^3 = 8 AHAAAA ima 8 MAD za svaki korak
Jel to onda znači da ako bi imali frame veći od ovog ne bi došli do ruba? Ako je step 3 uvijek na početku.
KUŽIM, TNX :D. Hvala purpurni! Ma ne, koristan je. Eventualno skratit.

meni nije jasna jedna stvar. kaze da s=s/2 i ponavljaj dok s!=1 zasto onda u svakom koraku imamo 8 MAD-ova

zato što on gleda sve okolo tipa

cemu onda s=s/2 ponavljaj sve dok s!=1 jasno, slijepa sam..hvala puno//pa za svaki korak do bloka udaljenosti 1 se računa po 8 MAD-ova, tj za prvi korak kad je S=3 se računa 9 jer se računa i za centralan, sve 5, nicht

Ako bi mi ulazne dimenzije bile npr. 16x16, onda bi mi step size na pocetku bio s=4? //s=2^3=8 i to znači računaš 8 mjera(btw vodite računa o tome da ne mora biti MAD), i onda je s=s/2=4 // Jasno je meni da računam 8 mjera. Samo kad imam sliku 16x16 mogu uzet 8 mjera za s=3 ili za s=4. S tim da ako uzmem za s=4, bit će slično pozicionirani blokovi kao kod 8x8 sa s=3(svi će dirati rubove frame-a).

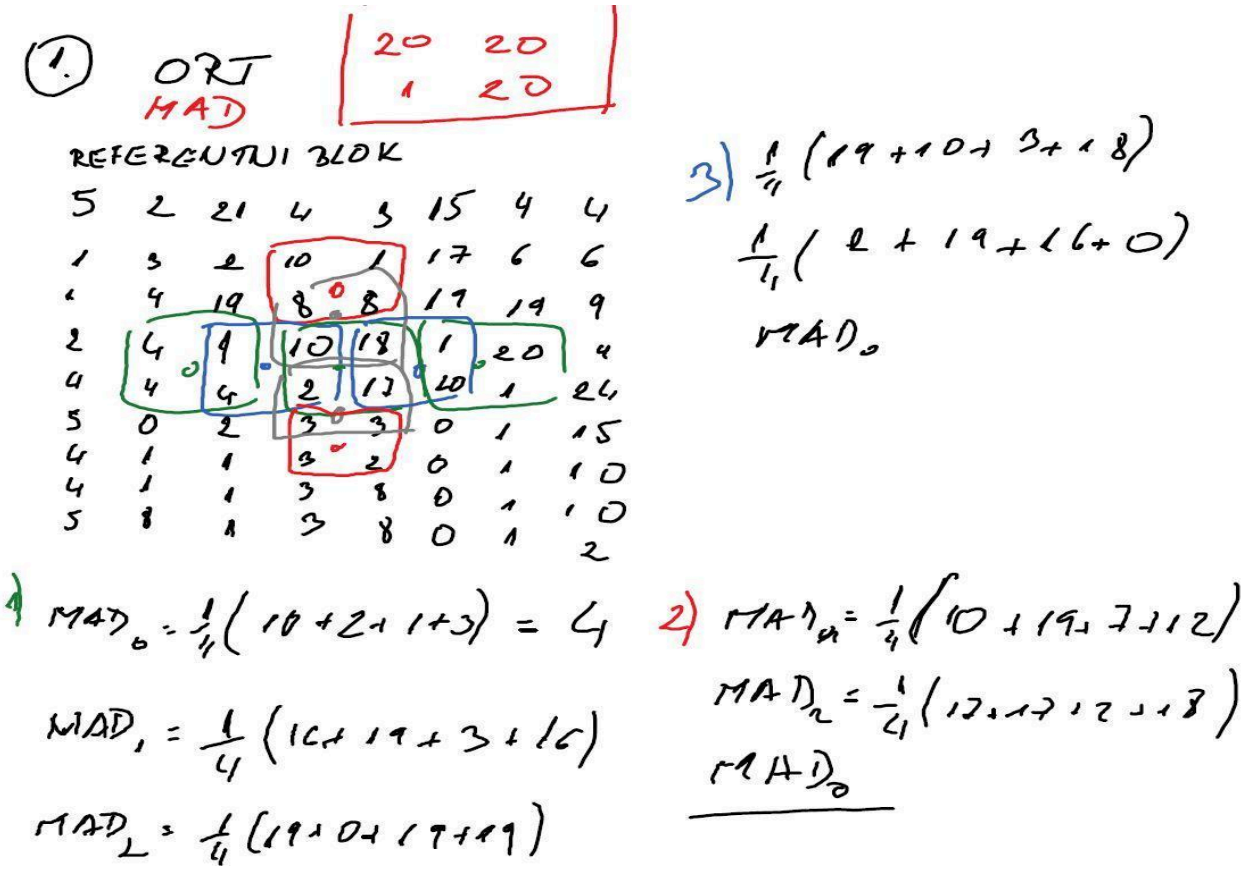
Prove me wrong, formula za s => (block_width)^s=(frame_width) // moze i height posto je kvadrat // moj bed, mozak mi je sjeban od brute forcanja ovog predmeta s=frame_width/block_width - 1 /a što potom zapravo gledaš kao frame_width//cijelu sliku, u gornjem primjeru je to 8, a block width je 2 //po tom bi ti onda za onu sliku sa slajda početni s bio 5? 12/2-1?// moze link na prezu + koji slajd //slajd 61, nemam pojma koja je preza printane su mi// nasao sam, ne znam kolika je velicina bloka tj. ne mogu odredit sa slike, iako izgleda ko da je 2x2. U tom slučaju ne štima formula. Ovdje samo kaže “An initial s ze is picked.” https://course.ece.cmu.edu/~ee899/project/deepak_mid.htm

EVO KOMENTAR KOLEGE OD PRETPROŠLE GODINE:

,

https://en.wikipedia.org/wiki/Block-matching_algorithm#Three_Step_Search -> jel ovo onda neki indijski način ako postavlja s=4 na početku? “Typical inputs are a macroblock of size 16 pixels and a search area of p = 7 pixels.”
https://www.youtube.com/watch?v=drDNVGKnVlY&t=12m29s

moze li netko malo bolje pojasniti onih 7 pozicija desno i 6 dolje kod full search-a iznad? → mreza je 8x8 ti imas 3x2 (ili 2x3) dimenzije prozora, zato se mozes pomicat 7 desno i 6 dolje (ili 6 desno i 7 dolje), nije provjereno!!!!
ne znam jesam stvarno toliki debil, ali dobijem 5 i 6; jel brojimo i nulti, odnosno da se ne micemo uopce?
//brojimo, jer se uspoređuju blokovi, ne znam kako lijepo formulirat to, ali da hah
mislim da sam skuzio sto si htio reci, hvala ti :) //nicht



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)MI

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?

Gledaj ko binarno traženje u 2D - kreneš od s/2, dijeliš s 2 nakon svake iteracije. To onda neće biti 3SS za blokove 16x16 i veće (jer tad ćeš imati sekvencu 8-4-2-1). A za neku divlju veličinu bloka, npr. 15x15 gledaš 15 // 2 = 7, 7 // 2 = 3, 3 // 2 = 1. // operator je cjelobrojno dijeljenje. Isto tako, za blokove 7x7 i manje ćeš imati 2SS (koraci 3 - 1 npr.). Asistent je spominjao da rješenja mogu degenerirati u 2SS ako je zadani blok premali.

Primijetiti pseudokod od ispod uzima 2^{n-1} za prvi korak, što će biti samo s/2 ali ceil toga za lijepe brojeve. Možete i tako ako vam je lakše, ali ne mijenja puno, samo će ta smanjivanja koraka biti uniformnija (jer će ići točno po potencijama broja 2). Oba načina smanjivanja koraka imaju mane

***** POGLEDATI KRAJ DjDžok-a**** =>**

STO JE d U OVIM PSEUDOKODOVIMA (kako se ra*evo ovdje (iduća stranica) da se ne mora skakati na kraj dokumenta*cuna)?

d se racuna po slici ispod ‘algoritam potpunog pretrazivanja’ na isti nacin za ort i log <- POTVRĐENO 2021.: potvrdio mag. ing. asistent Alen Duspara u mailu, d je udaljenost od ruba bloka do ruba prozora, tj. kao na donjoj slici

***ceka se potvrda/objasnjenje/tocna interpretacija sto je d* -> M a.. bar tak pise u objasnjenjima algoritama sa starih preza (ubacila sam slike ispod) ;sto bi ti onda rekla koliki je d u mi 2015 za ort? tj. jel tamo tocno izracunato?**

Nisam sigurna, al mislim da je d onda samo ono sta je zadano u zadatku, velicina prozora 4 (dakle d=4), samo je onda tamo krivo s izračunat (trebalo bi biti valjda s=(4+1) div 2 = 2) ispada da je po ovoj gornjoj slici isto 4

ovo gore ^ ...ne znam moze li se primjeniti na ove brze algoritme ali tu bi d bio broj pixela do referentnog bloka ako se ne varam? (sto bi znacilo da je u onom pdf-u rjesenje 2015. krivi korak -> umjesto 4 bi trebao biti 2)
Jel to sigurno da je tamo krivo? ;a ne znam, gledam u ovu sliku iznad i zbunila me tako da nisam sigurna.

Za buduce generacije/ispite: **tocno je da se d racuna po ovoj slici gore. ispod imate postupke za ORT i LOG gdje je d=8 i to je krivo, treba biti d=4 i onda je pocetni korak s=d/2=2**

23.01.2021 → što nije Hofman na zadnjem predavanju-rekao da je d širina/visina bloka???^^POGLEDATI PLAVI TEKST GORE, d = udaljenost ruba bloka do ruba prozora, slika iznad je točna

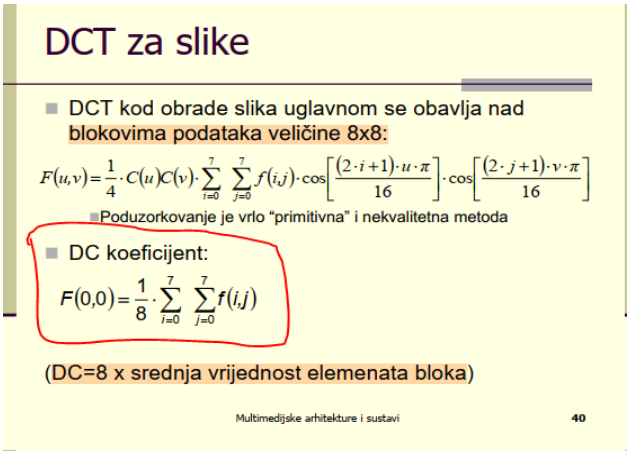
Jel onda za korak uzimamo (d+1)/2 ili d/2? - d/2, prema prezentacijama MAS-JK-3-5 slajdovi 21, 22, 23

|-> Ne znam je li smiješno ili žalosno da se nakon provedenog MI, nakon poslanih mailova i nakon toliko rasprave i dalje ne zna kako se određuje početni korak. Nositelji i asistenti različito govore, ne mogu se usuglasiti, na ispitu ne žele reći jer bi mi to trebali znati… Moguće je da je Hofman to rekao, ali na ispitu se ako se je uzelo d = polovica bloka dobilo točno rješenje pa ajmo se toga držati…… osim ako ne promijene sad fna Zulll jer eto nitko ne zna….

MI 2020/2021 (15 zadataka na zaokruzivanje; tocno 2b/netocno -0.2b)
-prvih 6 zadataka:

1. **log sa mse i ort sa mad**
   za svaki od ta dva algo:
   a) **b\poboljsanje brzine izvodenja za oba algoritma u odnosu na full search --kako se računa to poboljšanje brzine izvođenja? broj operacija u full search/broj u ovim algoritmima**
   b) **koliko iznosi mse ili mad u 2.koraku za desni blok**
   c) **konacni vektor pomaka za oba algoritma**

2. koji algoritam obavlja prvo horizontalno pa onda vertikalno : ort
3. koji algoritam ima konacan broj koraka neovisno o velicini prostora pretrazivanja : 3ss
4. par zadataka sa kvantizacijskim korakom i ulaznim nizom, sve na slicnu foru, kod jednog je trebalo kvantizirani q kodirati huffmanovim stablom i odrediti broj bitova tako kodiranog kvantiziranog ulaza
5. DC koeficijent kod DCT transformacije. matrica 8x8, f(i,j) = 4 za svaki i,j --- moze netko objasnit s primjerom?// samo zbrojimo sve elemente iz matrice i podijelimo s 8. Znaci na primjeru gdje je cijela matrica puna cetvorki dobijemo 64*4=256 i onda 256/8=32 pa je DC=32



6. **pretvorba RGB u YCbCr**

7. **zadaci s brzinom prijenosa (bilo je naznaceno u ispitu da 1MB = 2^20B)**

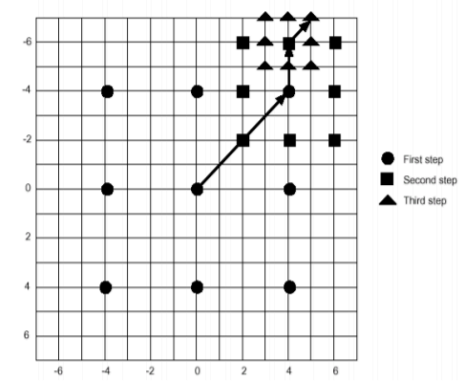
zna netko tocna rjesenja za LOG/ORT a)b)c)?

Sto ako u 3ss je vektor (0,0) vec najmanji, tj manji je od svih ostalih 8 novih koje sam nasao i prema kojim bi se trebao pomakuti? Jel nastavljam daljem (jer 3ss "UVIJEK" ima 25 koraka ili je to to)?

Nastavljaš dalje jer se smanjuje udaljenost, tj u idućem koraku računaš za blok koji ti je 'bliži' onda

- sam umjesto da gledas 8 tocaka oko neke tocke sa strane/gore/dolje, gledas tih 8 oko centra s pola koraka neg prethodno, ne?
- da

Pretraživanje u tri koraka (3SS)



0. N = 3
1. Compute SAE(0,0)
2. Set $S=2^{(N-1)}$ (step size)
3. Compute SAE on 2^N locs
4. Select loc with min SAE
5. Set S=S/2
6. Repeat [3:5] until S=1

- Svaki korak – 9 točaka za proračun
- Broj točaka: 9+8+8 = 25 (FS: 15*15 = 225)

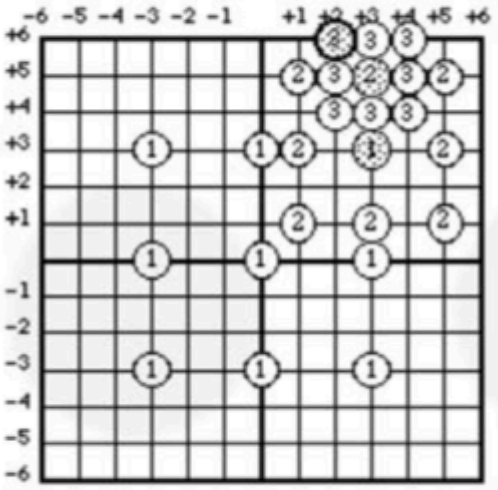
Three Step Search (TSS)

Ovaj algoritam je popularan zbog svoje jednostavnosti i robusnosti i vrlo dobrih performansi. Algoritam ima sljedeće korake:

Korak 1: Početna veličina koraka pretraživanja je 3. Osam blokova s udaljenošću koraka pretraživanja od središta pretraživanja se uspoređuju.

Korak 2: Veličina koraka se smanjuje za 1. Središte se pomiče na mjesto s minimalnom mjerom poremećaja.

Koraci 1 i 2 se ponavljaju dok korak pretraživanja ne bude jednak 1.



Problem koji se može javiti kod TSS algoritma je da koristi ravnomjerno razmještene točke provjere u prvom koraku, što može biti neučinkovito za malu procjenu pomaka.

ovo su 2 razlicita algoritma wat

17506023749357

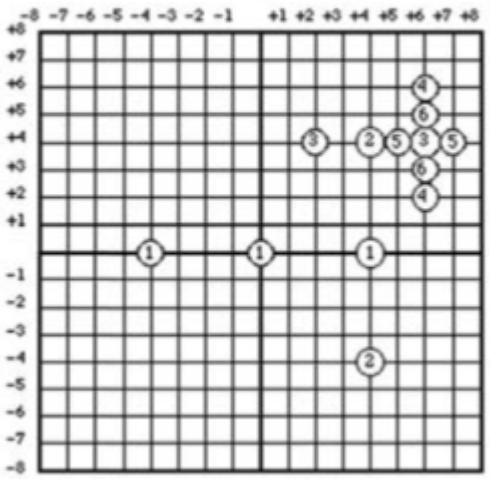
Orthogonal Search Algorithm (OSA)

Ovaj algoritam je hibrid TSS i TDL algoritama. Specifičan je po tome što najprije ispituje horizontalne blokove, a potom blokove u vertikalnom smjeru. Algoritam ima sljedeće korake:

Korak 1: Početna veličina koraka se računa kao (d +1) div 2, gdje je d veličina prozora pretraživanja. Uspoređuju se dva bloka na udaljenosti od koraka pretraživanja u vodoravnom smjeru od središta (i središte) te novo središte postaje (ili ostaje) blok s najmanjom mjerom poremećaja.

Korak 2: Potom se to isto radi u vertikalnom smjeru.

Korak 3: Prepoloviti veličinu koraka pretraživanja, te ponavljati sve dok je korak pretraživanja veći od 1.

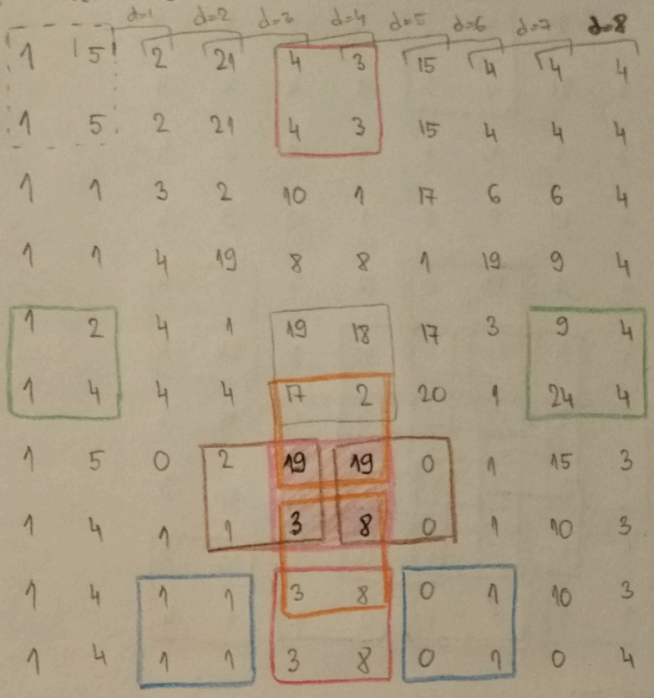


ZADATAK 4

tekući blok:

20	20
1	20

referentni blok:



d = max displacement

ORT: $S = (d+1)/2 = 4$

MSE

veličina prozora = 4

(1) $MSE_1 = \frac{1}{4}\left[(20-19)^2 + (20-18)^2 + (1-17)^2 + (20-2)^2\right] = 146,25$

(2) S = 4 horizontalno

lijevo: $MSE_2 = \frac{1}{4}\left(19^2 + 18^2 + 0^2 + 16^2\right) = 235,25$

desno: $MSE_2 = \frac{1}{4}\left(11^2 + 16^2 + (-23)^2 + 16^2\right) = 290,5$

min: MSE_1

(3) S = 4 vertikalno

gore: $MSE_3 = \frac{1}{4}\left(16^2 + 17^2 + (-3)^2 + 17^2\right) = 210,75$

dolje: $MSE_3 = \frac{1}{4}\left(17^2 + 12^2 + (-2)^2 + 12^2\right) = 145,25$

min: MSE_3 dolje

(4) S = S/2 = 2 horizontalno

lijevo: $MSE_4 = \frac{1}{4}\left(19^2 + 19^2 + 0^2 + 19^2\right) = 270,75$

desno: $MSE_4 = \frac{1}{4}\left(20^2 + 19^2 + 1^2 + 19^2\right) = 280,75$

min: MSE_3 dolje

(5) S = 2 vertikalno

$MSE_5 = \frac{1}{4}\left(1^2 + 1^2 + (-2)^2 + 12^2\right) = 37,5$

min: MSE_5

(6) S = S/2 = 1 horizontalno

lijevo: $MSE_6 = \frac{1}{4}\left(18^2 + 1^2 + 0^2 + 17^2\right) = 153,5$

desno: $MSE_6 = \frac{1}{4}\left(1^2 + 20^2 + (-7)^2 + 20^2\right) = 212,5$

min: MSE_5

(7) S = 1 vertikalno

gore: $MSE_7 = \frac{1}{4}\left(3^2 + 18^2 + (-18)^2 + 1^2\right) = 164,5$

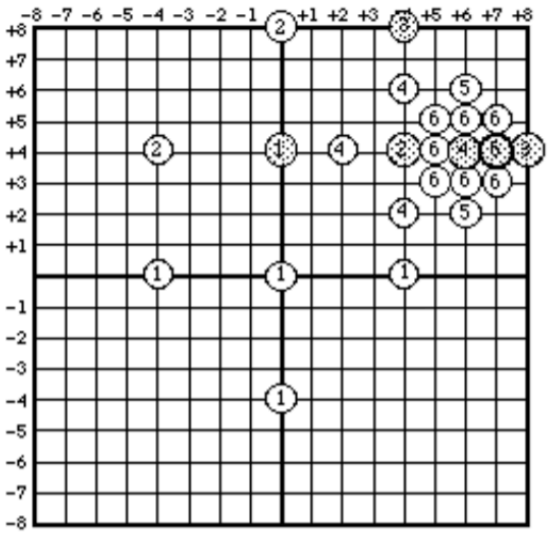
dolje: $MSE_7 = \frac{1}{4}\left(17^2 + 12^2 + (-2)^2 + 12^2\right) = 145,25$

min = MSE_5

vektor pomaka: 0, -2

broj pretraživanja: 12

Logaritamsko pretraživanje (LOG)



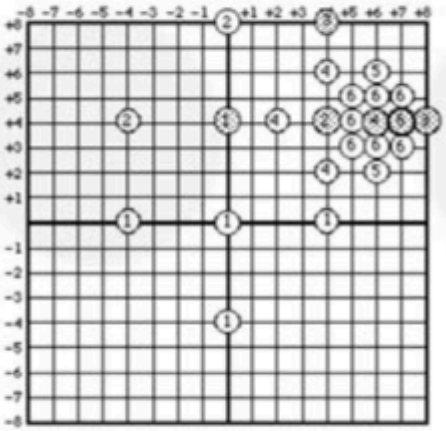
0. Step size = S (d/2) = 4
1. Search (0,0)
2. Search 4 loc S pixels away (horiz and vert.)
3. New origin = best match (BM)
4. If BM is in center: S = S/2
5. If S=1: goto 6, else 2
6. Search 8 loc arround BM and select BM

Broj pretraživanja: Ovisi o podacima

OBJ

OBJ

Two Dimensional Logarithmic Search (TDL)

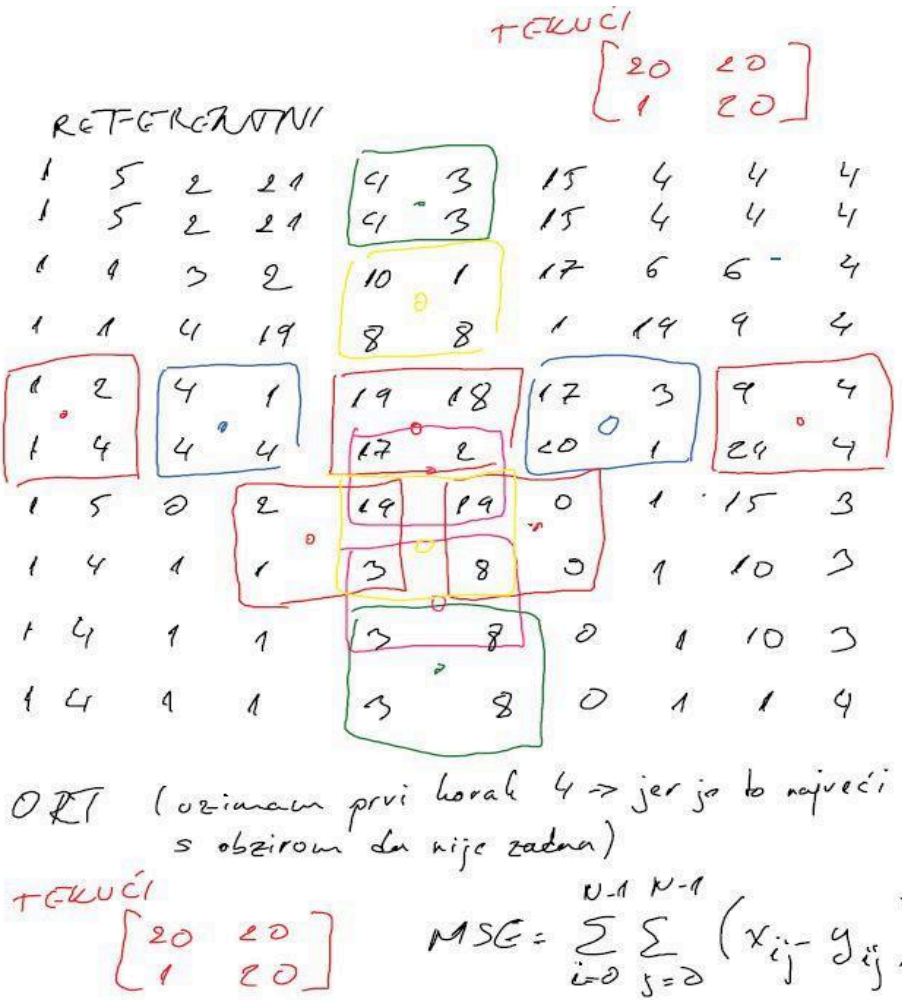


Ovaj algoritam zahtijeva više koraka nego TSS, te je u principu precizniji, pogotovo kada je prozor pretraživanja velik. Algoritam ima sljedeće korake:

Korak 1: Ako je prozor pretraživanja d, onda se početni korak pretraživanja računa kao $2^{\lfloor \log_2 d \rfloor - 1}$ Uspoređuju se 4 bloka n udaljenosti koraka pretraživanja od središta i središte pretraživanja.

Korak 2: Ako je najmanja mjera poremećaja u središtu, prepoloviti veličinu koraka. Ako nije, novo središte postaje točka s najmanjom mjerom poremećaja i ponavlja se korak 1.

Korak 3: Kad veličina koraka postane 1, svih devet blokova oko središta se uspoređuju i odabire se onaj s najmanjom mjerom poremećaja.



//tu ispod su riješena mislim dva različita zadatka, za ovaj lijevi je s=2, a za desni je s=4, obrat 1 i pažnju i da piše tekući

13. MI: 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

14. MI: 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
- 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 vrijednosti za dvije različite komponente, kako bi se kompresija napravila kako spada, bez gubitka bitnih podataka.

15. MI: 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 do neprepoznatljivosti. Ako je nakon DCT-a obradiš , možeš reći koje frekvencije ćeš isfiltrirati da je slika i dalje razumljiva i prepoznatljiva.

MI: Objasniti zašto se vrijednosti slikovnih elemenata umanjuju za 2^{n-1} (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.**

16. MI: Definirati mjere poremećaja pri izračunu vektora pokreta

$$MSE(x,y;d_x,d_y)=\frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\left[Y(x+i,y+j,t_1)-Y(x+d_x+i,y+d_y+j,t_0)\right]^2$$

$$MAD(x,y;d_x,d_y)=\frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}\left|Y(x+i,y+j,t_1)-Y(x+d_x+i,y+d_y+j,t_0)\right|$$

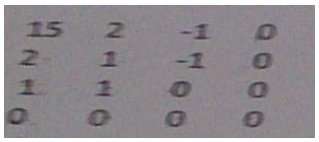
$$MPC(x,y;d_x,d_y)=\frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}T\left[Y(x+i,y+j,t_1),Y(x+d_x+i,y+d_y+j,t_0)\right]$$

$$T(\alpha,\beta)=\begin{cases}1, & \text{za } |\alpha-\beta|\le T_0\\ 0, & \text{inače}\end{cases}$$

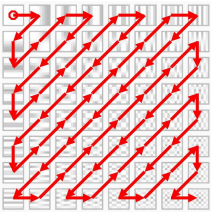
Može netko malo pojasniti kako se točno računa ova treća mjera?

i. 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)

17. MI: 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 //tocno je 128 da.***
// stupanj kompresije = 00
*pocetni niz imao je 16 brojeva*8 bita / (pobrojis sve nule i jedinice koje si dobio u novom kodu)*
// kako se računa amplituda za negativne brojeve?
Pretpostavimo da je koeficijent C zapisan u formatu dvojnog komplementa, a K je kategorija kojoj taj koeficijent pripada. • Ako je C pozitivan broj tada će se Huffmanovom kodu, kao proširenje, dodati K nižih bitova od C. • Ako je C negativan tada će se Huffmanovom kodu dodati K nižih bitova od vrijednosti koeficijenta C umanjenog za jedan, tj. (C-1). (2. prezentacija, slajd 75.)

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

HUFFMAN AC

Duljina niza/Kategorija	Duljina koda	Kodna riječ
0/0 (EOB)	4	1010
0/1	2	00
0/2	2	01
0/3	3	100
0/4	4	1011
0/5	5	11010
0/6	7	1111000
0/7	8	11111000
0/8	10	1111110110
0/9	16	1111111110000010
0/10	16	1111111110000011
1/1	4	1100
1/2	5	11011

HUFFMAN DC

Kategorija	Duljina koda	Kodna riječ
0	2	00
1	3	010
2	3	011
3	3	100
4	3	101
5	3	110
6	4	1110
7	5	11110
8	6	111110
9	7	1111110
10	8	11111110
11	9	111111110

KATEGORIJA

Kategorija	DC razlika	AC koeficijent
0	0	
1	-1,1	-1,1
2	-3,-2,2,3	-3,-2,2,3
3	-7,..,-4,4,..,7	-7,..,-4,4,..,7
4	-15,..,-8,8,..,15	-15,..,-8,8,..,15
5	-31,..,-16,16,..,31	-31,..,-16,16,..,31
6	-63,..,-32,32,..,63	-63,..,-32,32,..,63
7	-127,..,-64,64,..,127	-127,..,-64,64,..,127
8	-255,..,-128,128,..,255	-255,..,-128,128,..,255
9	-511,..,-256,256,..,511	-511,..,-256,256,..,511
10	-1023,..,-512,512,..,1023	-1023,..,-512,512,..,1023
11	-2047,..,-1204,1024,..,2047	

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

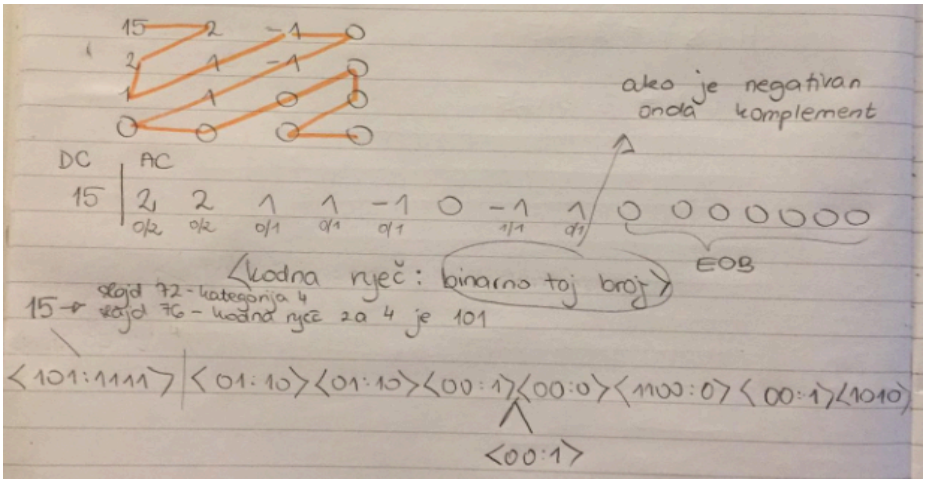
Budući da nema nijednog drugog bloka, 15 se ne mora od niceg oduzet
stoga je 15-0 = 15 i DC je 15 // slajd 70, 2. prezentacija i 20. slajd kompresija slika
15 spada prema tablici (DC) u kategoriju(broj bitova potrebnih za binarni zapis broja) 4 (jer se moze zapisati sa 4 bita), za kategoriju 4 kodna riječ je 101, Dakle slijedi

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

// kako to da kodiramo 15 binarno, a ne pomak unutar kategorije, odnosno 7 (111) ? ("*Kako se kategorijom ne može točno odrediti vrijednost elementa poslije kategorije mora se poslati još podatak [amplituda] koji unutar kategorije definira koji je to element.*" - slajd 72). Aha vidim i da kaze da amplituda ima tolko bitova kolki je redni broj kategorije.
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 ovo 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 - ovo je onak 100% ne objasnjenje pitanja, pitanje je kak znamo koja je kategorija broja, za DC je bilo po duljini binarnog zapisa, a kak je za AC? Samo taj broj?
za AC kategoriju gledas isto kao za DC, duljina binarnog zapisa

2 -> <01,10>
1 -> <00,1> //0/1 ...0 nula ispred jedinice i kategorija 1 kodna riječ 00, binarni zapis je 1
1 -> <00,1> // 0/1 kodna riječ 00 i
-1 je komplement(1)-> <00,0> ?

//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 računamo
-1 -> <1100:0> //Sada je kodna riječ 1100 jer je jedna nula ispred -1, kategorija 1 i onda je 1/1->1100, binarni zapis je 0 zbog komplement(1)// di pisu ta sva pravila? kakve su ovo magije? //tablice na slajdu 76-77//odakle se da zakljucit da ak je 0 ispred ide 1100:binarni_broj//-1 je kategorija 1 po tablici(72slide) , onda gledas tablicu(77slide) i redak di pise 1/1 (1je jer imas 1 nulu ispred, 1 je jer je kategorija 1) i to ti je kod 1100…. binarni broj je 0 jer je to komplement od binarne jedinice (zbog minusa)//kuzim taj dio s komplementom, bunilo me samo zbog cega ide 1100 na pocetku, ugl tenks skuzio sam :)
1 -> <00,1>
I još 7 nula
1010 EOB(end of block)
// Zašto 1010? Koja je logika iza tog broja? - Guglao sam, izgleda da je 1010 jednostavno kod za EOB
Dakle dobijemo od 16 brojeva (128b) 36b kodiranih, kompresije 128/36 = 3.56.
26.1.2021 - Otkud 128b za ne kodirani niz??????? Što ne bi trebalo biti 4 (broj znamenki za 15 - 1111) puta (4 x 4) - dimenzije matrice = 64??
//Imamo 16 brojeva i za zapis svakog nam treba 8 bitova(jer je raspon -128 do 127) pa 16*8=128
Kako se dobije ovih 36b? Što trebam brojiti?
//brojiš binarne znamenke u konačnom zapisu npr <1100:0> je 5 znamenki
Kako bi zapisali da imamo zadano -2?
//ugl da imas broj -2, prvo gledas u tablici kategorija i vidis da je -2 kategorija 2.. recimo da u zig-zag nizu nema nula ispred tog -2 onda je to 0 nula i uz kategoriju 2 imas 0/2 .. sad to 0/2 gledas u tablici za AC simbole i vidis da je to kodna rijec 01 (to ce ici u prvi dio onog <_:_> zapisa).. 2 binarno je 10, ali posto je -2 uzimas komplement odnosno 01 i dobivas <01:01>.. bar sam ja tako shvatila)



4

Mislim da je ova rasprava dovoljna za shvatit sliku, treba se samo znati snaći u korištenju tablica, s razumijevanjem prateći tablice pročitati raspravu i skuži se

ulaz: kvantizacijski niz [x1,x2,x3], DC(i-1)
DC koeficijent:
DC = DC(i) - DC(i-1)
DC = x1 - DC(i-1) [zadano ili 0]
kodirano kao: <HUFF_dc(kat(DC)) : 1C(x1)>
HUFF_dc je tablica kodiranih kategorija
kat(DC) je log_2(DC) tj. min broj bitova za zapis DC u binarnom
1C je 1-komplement tj bit-flipanje ako je broj negativan
AC koeficijent:
x2 (isto i za sve ostale; 0 se preskače)
kodiran kao: <HUFF_ac(N(0,x2)/kat(x2)) : 1C(x2)>
HUFF_ac je tablica kodiranih simbola 'N(0,x)/kat(x)'
N(0,x) je broj nula ispred broja x
izlaz: <HUFF_dc(kat(DC)) : 1C(x1)><HUFF_ac(N(0,x2)/kat(x2)) : 1C(x2)><HUFF_ac(N(0,x3)/kat(x3)) : 1C(x3)>

primjer: https://www.sciencedirect.com/topics/engineering/huffman-table
ulaz: kvantizacijski niz [42 16 -21...], DC(i-1) = 40
1. DC = 42 - 40 = 2
2 kat(2) = 2 -> HUFF_dc(2) = 011
1C(2) = 10
kod: <011 : 10>

2. 16
kat(16) = 5 (jer 16(dec) = 10000(bin) -> 5 bitova)
N(0,16) = 0
simbol = '0/5'
HUFF_ac(0/5) = 11010
1C(16) = 10000
kod: <11010 : 10000>

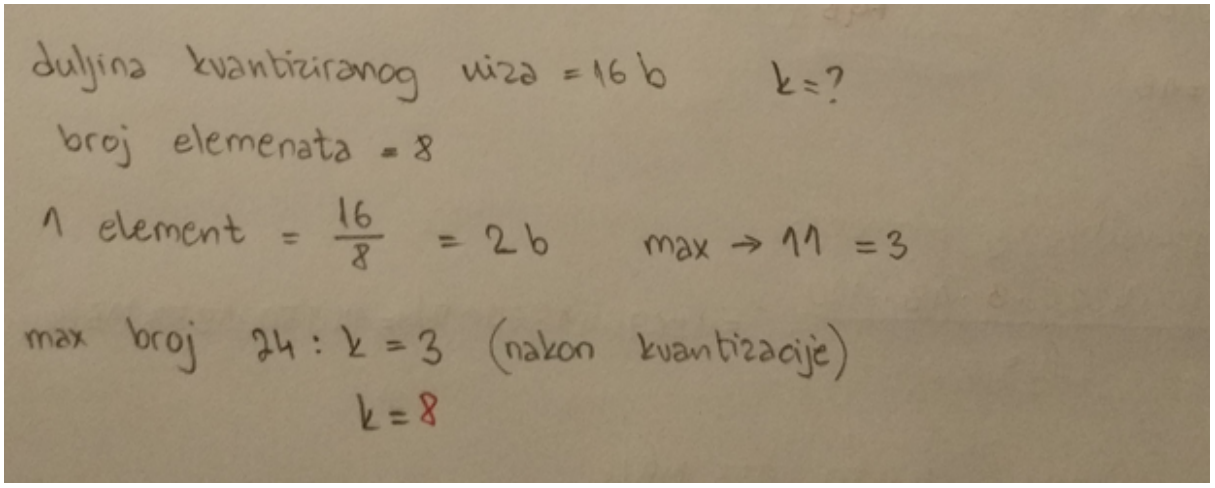
3. -21
kat(-21) = 5 (jer 21(dec) = 10101(bin) -> bitova)
N(0,-21) = 0
simbol = '0/5'
HUFF_ac(0/5) = 11010
1C(-21) = bit-flip(10101) = 01010
kod: <11010 : 01010>
izlaz: 011 10 11010 10000 11010 01010

18. MI: Za 8K UHDTV video 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)

Točno rješenje: 10080*4320*120*3 B = 15 676416000 B / 10^6 = 15676.416 MB/s
Stupanj kompresije: 15676.416 / 37.45 = 418.5959

19. MI: Za postupak kvantizacije nad ulaznim nizom 0, 5, 10, 16, 24, 4, 5, 8 uz kvantaizcijski korak 5 odrediti:

- **Koliko je minimalno bitova potrebno za zapis ulaznog nekvantiziranog niza?**
  i. najveći broj je 24 koji se može zapisat u 5 bitova jer 2^5 = 32 (alternativno ceil(log2(24)))
  ii. 5 * 8 = 40 bitova za čitav niz
- **Koliko je minimalno bitova potrebno za zapis izlaznog kvantiziranog niza q u slučaju zaokruživanja kvantizirane vrijednosti ceil [q] i u slučaju zaokruživanja floor [q]?**
  i. q = 5, slučaj ceil(q) -> 0,1,2,4,5,1,1,2 -> potrebno je min 3 bita, 3* 8 = 24 bita za niz
  ii. q = 5, slučaj floor(q) -> 0,1,2,3,4,0,1,1 -> potrebno min 3 bita, 3*8 = 24 bita za niz
- **Odredite restaurirane nizove za oba slučaja zaokruživanja?**
  i. slučaj ceil(q) -> 0,5,10,20,25,5,5,10
  ii. slučaj floor(q) -> 0,5,10,15,20,0,5,5
- **Odredite kvantizacijski korak tako da duljina kvantiziranog niza bude 16 bitova (za oba slučaja zaokruživanja) ?**
  *Napomena: max -> 11 (binarno) -> 3 (decimalno)



Prilično sam siguran da je k=8 samo u slučaju ceil(), a u slučaju floor možemo uzeti i k=7.
Prema tome za ceil() i k=8 dobivamo niz [0, 1, 2, 2, 3, 1, 1, 1], a za floor() i k=7 dobivamo [0, 0, 1, 2, 3, 0, 0, 1] (i dalje je 3 najveći broj, dakle stane u 2 bita).
Ipak, zbog elemenata 8, 16 i 24 rekonstrukcija je bolja kada se koristi k=8:

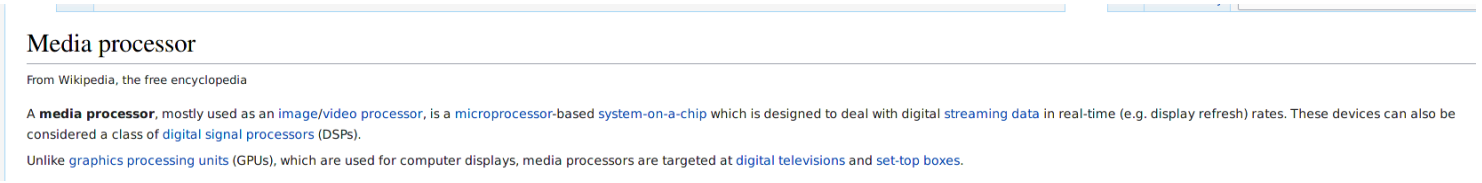
floor(), k = 8	[0, 0, 8, 16, 24, 0, 0, 8]
floor(), k = 7	[0, 0, 7, 14, 21, 0, 0, 7]
izvorno	[0, 5, 10, 16, 24, 4, 5, 8]

20. MI: Objasnite svojstva i međusobne razlike između medijskog koprocesora i medijskog procesora?

- **Medijski procesor**
  i. 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,....
  ii. 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**
  i. posebni procesor koji se stavlja u sustav da bi se u njemu obrađivali zahtjevni dijelovi algorit

ii. ma.
iii. 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, ...)

- *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.~~



Media processor

From Wikipedia, the free encyclopedia

A **media processor**, mostly used as an image/video processor, is a microprocessor-based system-on-a-chip which is designed to deal with digital streaming data in real-time (e.g. display refresh) rates. These devices can also be considered a class of digital signal processors (DSPs).

Unlike graphics processing units (GPUs), which are used for computer displays, media processors are targeted at digital televisions and set-top boxes.

Ovo gore je netočno: grafička kartica je medijski koprocesor jer bez procesora ne može obavljati svoje zadatke (obje konente komuniciraju). Medijski procesor bi bio SoC u TV prijemniku ili TV-u.

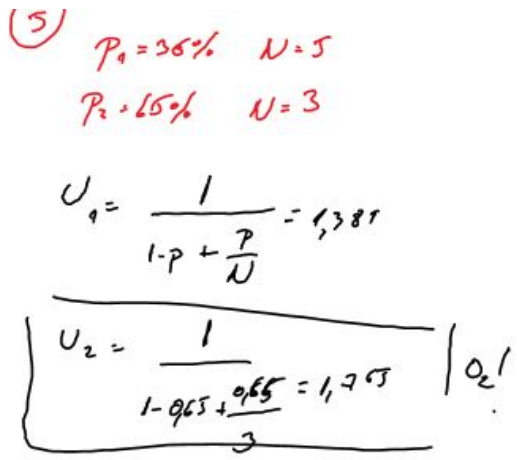
MI 2020.

U 12. pitanju bio je zadani ulazni niz [18, 6, 17, 21, 11, 27, 3]. Prstupak kvantizacije se izvodio sa zaokruživanjem prema nižem broju s korakom 7. Onda je kvantizirani niz [2, 0, 2, 3, 1, 3, 0] (podijeljeno s 7 I zaokruženo prema dolje]. Restaurirani niz se dobije množenjem kvantiziranog niza s kvantizacijskim korakom što je u ovom slučaju [14, 0, 14, 21, 7, 21, 0].

U 14. pitanju tražilo se koliko je minimalno bitova potrebno za zapis cijelog kvantiziranog niza. Ulazni niz je bio [43, 177, 61, 12, 129], a korak 11. Koristi se binarno kodiranje.
U tom slučaju kvantizirani niz je [3, 16, 5, 1, 11]. Kada ne koristimo statističke metode kodiranja, širina bitova svake zapisane vrijednosti mora biti ista. To znači da tražimo vrijednost u kvantiziranom nizu za koju treba najviše bitova. To je u ovom slučaju 16 za koju treba 5 bitova. U nizu je 5 vrijednosti što znači da je odgovor 5 x 5 bitova = 25 bitova.

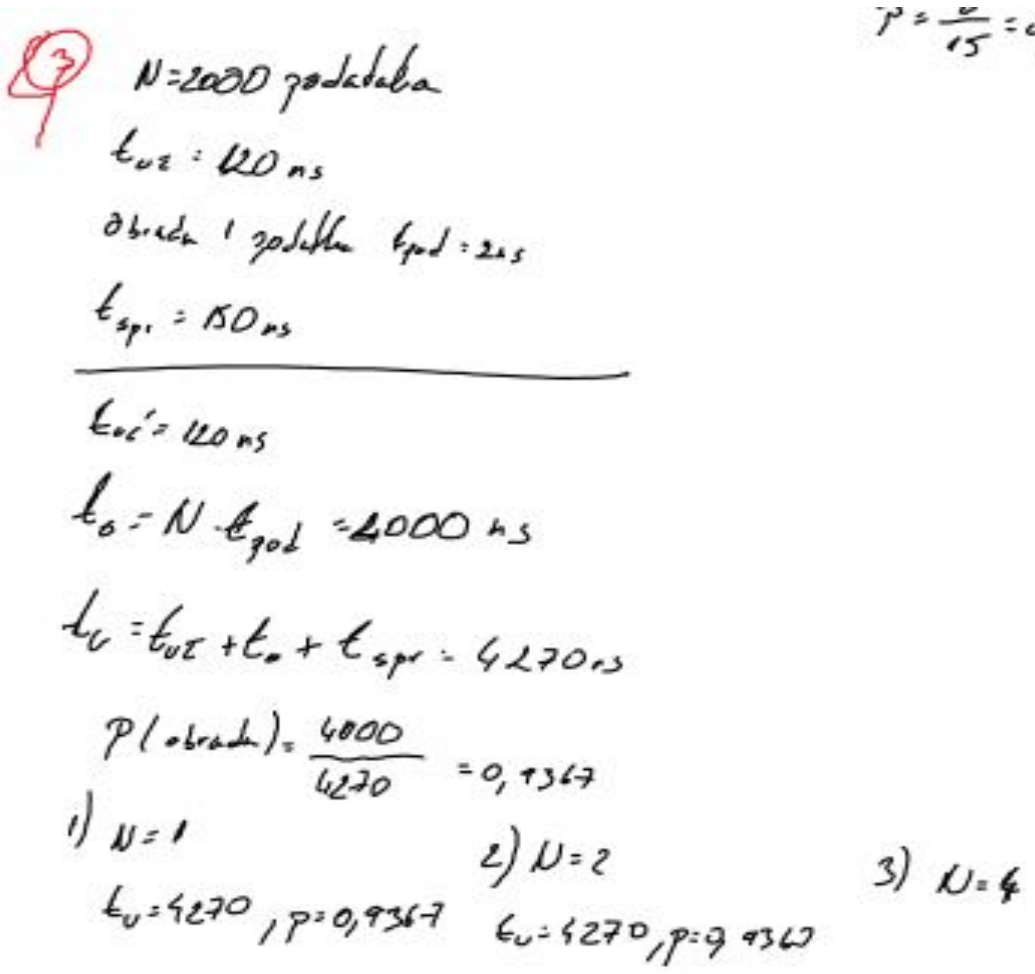
ZAVRŠNI ISPIT

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?

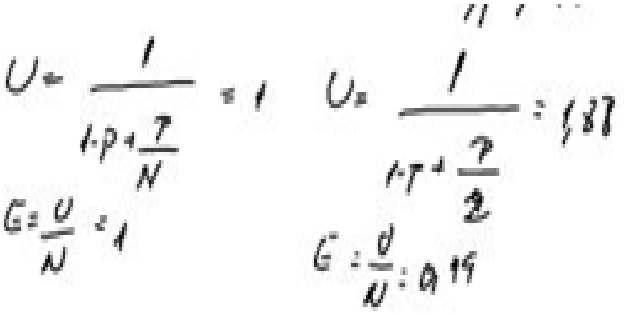


Zna netko šta je N a šta U?1 ->
// **N** -> broj puta koliko se može ubrzati izvođenje potprograma
// **U** -> ubrzanje programa
// **p** -> postotak, vrijeme izvođenja (dio programa **O** u omjeru **p** se može ubrzati **N** puta)
U = 1/(p1/N1 + p2/N2) - računa se po toj formuli (barem piše na fer2)
//zapravo bi trebalo biti U = 1 / (1 - p1 - p2 + p1/N1 + p2/N2), al 1 - p1 - p2 = 0 pa ispadne //gornja formula
U = 3.488 - što je ovo? ubrzanje uz obje optimizacije

21. **ZI 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.**



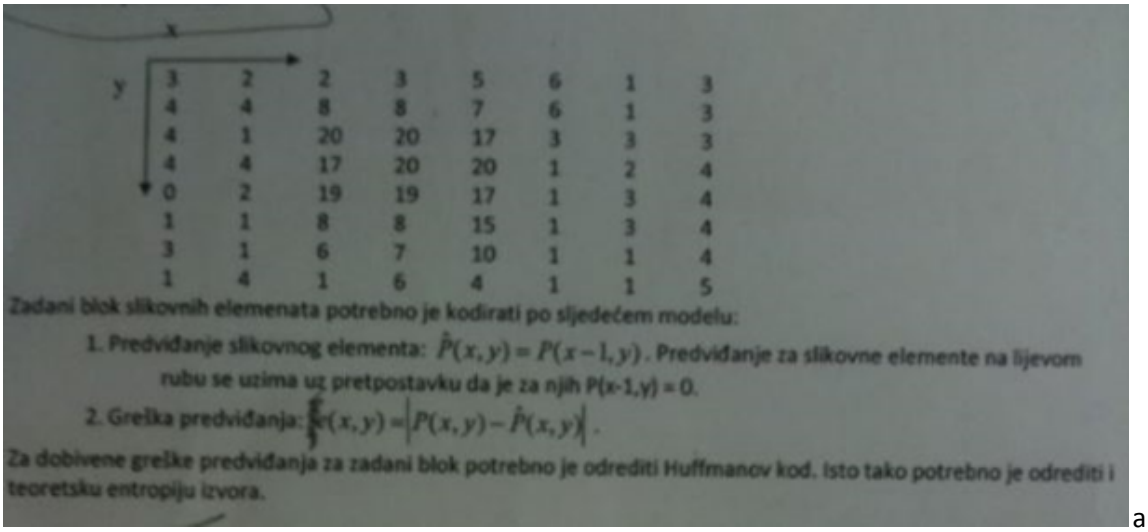
p = 0.9367 i t = 4270 za sva tri slučaja, samo se N mijenja (1/2/4)
//zar ne bi trebalo biti t0 =4270, t1 = 2270 i t2 = 1270 (analogno za p-ove)? ja isto mislim
// ne bi trebalo bikajti, vrijeme ukupno se izračuna jednom i onda se izračuna
// Vrijeme koje paraleliziraš paraleliziraš tako da ga uzmeš onakvo kakvo je u serijskom sustavu (1 procesor).



U=1/(1-P+P/N)
E=U/N
za 2 procesora U = 1.88, E = 94% -> TO JE TO!
za 4 procesora U = 3.3622, E = 84% -> TO JE TO! -bravo

22. ZI? MI: Tekst na slici

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 huffm



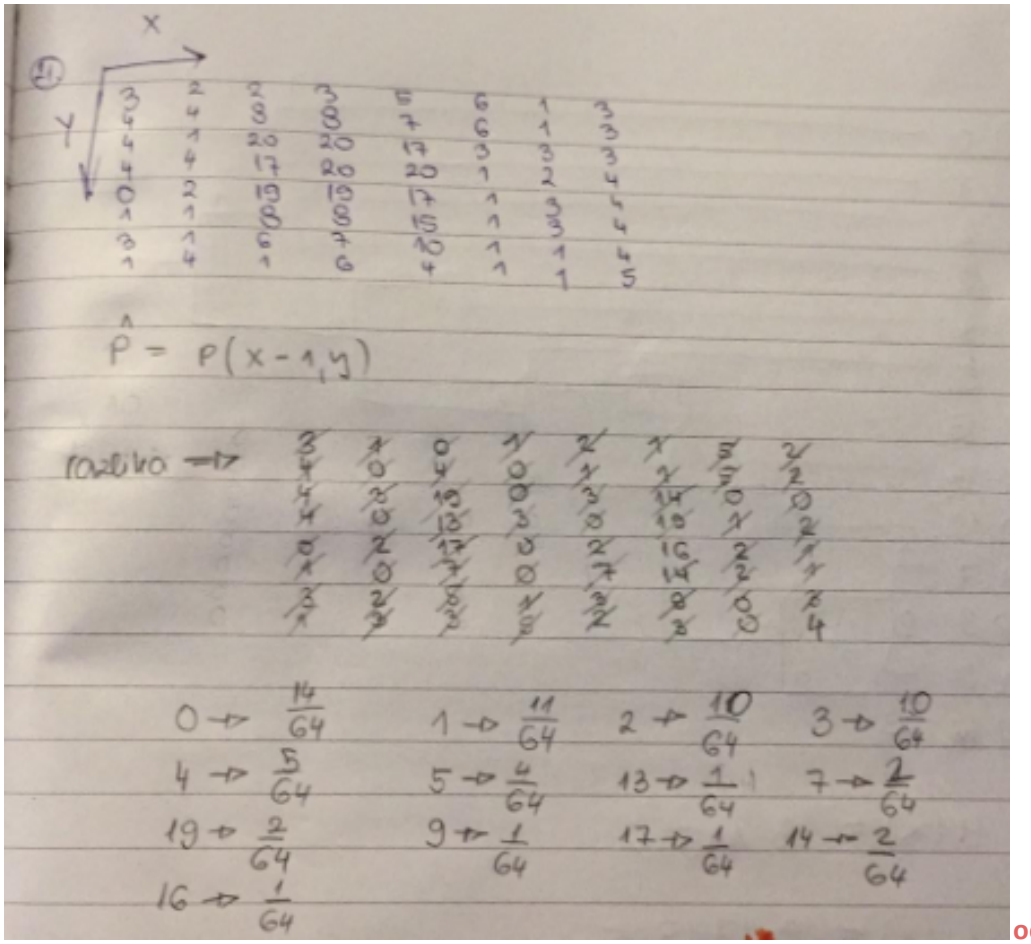
y \ x							
3	2	2	3	5	6	1	3
4	4	8	8	7	6	1	3
4	1	20	20	17	3	3	3
4	4	17	20	20	1	2	4
0	2	19	19	17	1	3	4
1	1	8	8	15	1	3	4
3	1	6	7	10	1	1	4
1	4	1	6	4	1	1	5

Zadani blok slikovnih elemenata potrebno je kodirati po sljedećem modelu:

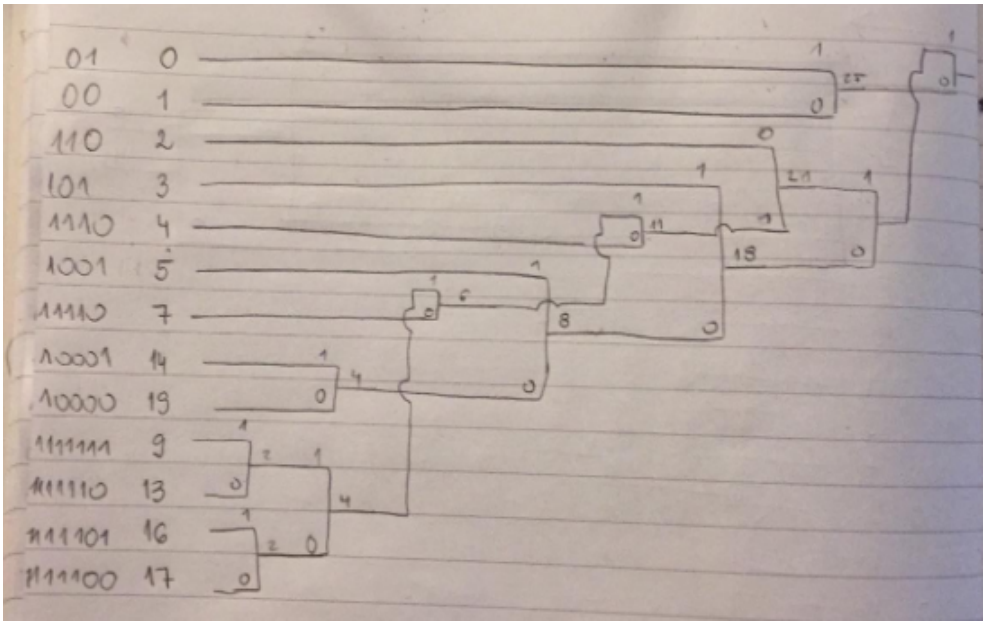
1. Predviđanje slikovnog elementa: $\hat{P}(x,y) = P(x-1,y)$. Predviđanje za slikovne elemente na lijevom rubu se uzima uz pretpostavku da je za njih P(x-1,y) = 0.
2. Greška predviđanja: $e(x,y) = |P(x,y) - \hat{P}(x,y)|$.

Za dobivene greške predviđanja za zadani blok potrebno je odrediti Huffmanov kod. Isto tako potrebno je odrediti i teoretsku entropiju izvora.

an.



ocemo proc?neucemo rip



H(X)=3.1313, L(X)=3.1875, MISLIM DA JE DOBRO TO -> dobro je, provjereno

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

24. ZI 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)

- 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.

25. ZI Koji su osnovni problemi koji su uzrokovali zastoj u povećanju performansi jednojezgrenih računalnih arhitektura?

http://superuser.com/questions/152011/why-multi-core-processors

- Pregrijavanje, manje cache memorije dostupno ili je potrebna jako velika memorija što zauzima puno prostora na die-u, usporen rast i efikasnost.

26. ZI: 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
- Nisam siguran. Bubnuo bih da bi to trebala biti veličina L1 cachea kako bi procesor minimalno čitao iz L2 ili dalje. Ispravite me…
- ^ Ne vidim kakve veze ima cache kod SIMD naredbe? SIMD funkcionira na način da ima veće registre (znači umjesto 32/64b ima recimo 128/256 ili 512b (npr AVX-512). U registar smjestimo vektor vrijednosti i SIMD ih obradi paralelno. Čini mi se da je tu najbolji odgovor da je optimalna širina 8b jer je dobar omjer kvalitete slike i efikasnosti. Za AVX-512 onda možemo paralelno obraditi 64 komponente, znači ~21 piksel po operaciji.

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

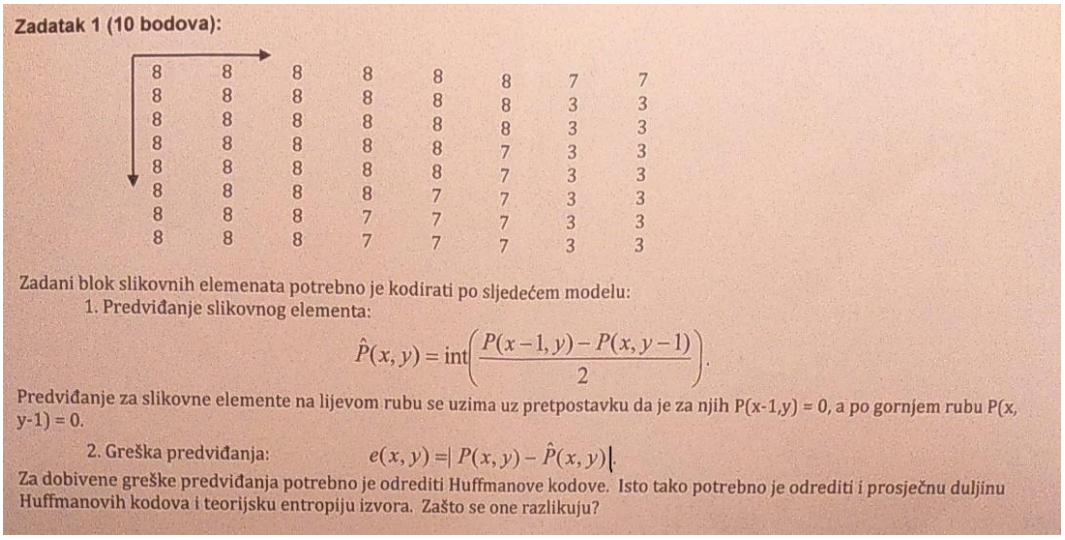
- (MAS-JK-3-4, slajdovi 7 i 8)
- Deadlock refers to a situation when a thread waits for an event that never occurs. This is usually the result of two threads requiring access to resources held by the other thread.
- Livelock refers to a situation when threads are not making progress on assigned computations, but are not idle waiting for an event.
- Livelock je kada dva procesa rade i mijenjaju, ali ne mogu se odlockati, a 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.
- https://stackoverflow.com/questions/6155951/whats-the-difference-between-deadlock-and-livelock

28. iZI Usporedite korake te prednosti i nedostatke u JPEG i Wavelet kompresiji.

- Koraci: JPEG: RGB u YUV, pomak, 2DCT, kvantizacija
- 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 najviš o i 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. ZI? MI: // otkud vam da je ovo MI? meni izgleda kao *zavrsni!*



Zadatak 1 (10 bodova):

8	8	8	8	8	8	7	7
8	8	8	8	8	8	3	3
8	8	8	8	8	8	3	3
8	8	8	8	8	7	3	3
8	8	8	8	8	7	3	3
8	8	8	8	7	7	3	3
8	8	8	7	7	7	3	3
8	8	8	7	7	7	3	3

Zadani blok slikovnih elemenata potrebno je kodirati po sljedećem modelu:

1. Predviđanje slikovnog elementa:

$$\hat{P}(x,y) = \text{int}\left(\frac{P(x-1,y) - P(x,y-1)}{2}\right).$$

Predviđanje za slikovne elemente na lijevom rubu se uzima uz pretpostavku da je za njih $P(x\text{-}1,y) = 0$, a po gornjem rubu $P(x, y\text{-}1) = 0$.

2. Greška predviđanja: $e(x,y) = |P(x,y) - \hat{P}(x,y)|$.

Za dobivene greške predviđanja potrebno je odrediti Huffmanove kodove. Isto tako potrebno je odrediti i prosječnu duljinu Huffmanovih kodova i teorijsku entropiju izvora. Zašto se one razlikuju?

(upitnike na sici zanemarit) A KAK SMO DOBILI 0. STUPAC P KAPA? kaže ti da je po lijevom rubu P(x-1,y)=0 pa onda imaš int(-8/2) Bravo, kapa dole :)

Jel onda ovo MI ili nije lmao? Fuck knows. Entropija i Huffman su prvi dio predavanja, no ne sjecam se greski redvidjanja igdježžč

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

H(X)=2.405 L(X)=2.5

$$H[X] = -\sum_{i=1}^{n} P(x_i) \log P(x_i)$$

$$H(X) = \sum_{i=1}^{n} P(x_i) \log_2\left(\frac{1}{P(x_i)}\right)$$

=

$$L(X) = \sum_{i=1}^{n} P(x_i) \cdot l_i$$

X →

30. Y ↓

8	8	8	8	8	8	7	7
8	8	8	8	8	8	3	3
8	8	8	8	8	8	3	3
8	8	8	8	8	7	3	3
8	8	8	8	8	7	3	3
8	8	8	8	7	7	3	3
8	8	8	7	7	7	3	3
8	8	8	7	7	7	3	3

$$\hat{P}(x,y) = \text{int}\left(\frac{P(x-1,y) - P(x,y-1)}{2}\right)$$

$$e(x,y) = |P(x,y) - \hat{P}(x,y)|$$

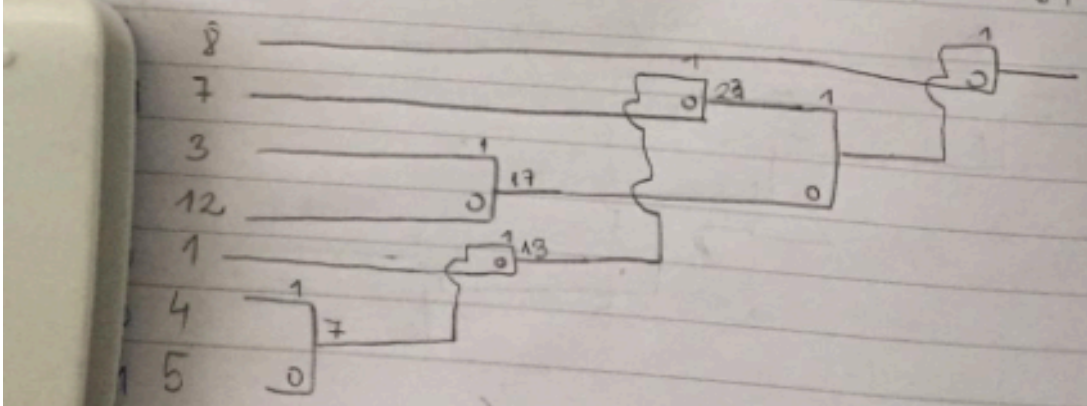
$\hat{P}$ →

0	4	4	4	4	4	4	3
-4	0	0	0	0	0	0	-2
-4	0	0	0	0	0	0	0
-4	0	0	0	0	0	0	0
-4	0	0	0	0	0	0	0
-4	0	0	0	0	0	0	0
-4	0	0	0	0	0	0	0

rozlika →

8	4	4	4	4	4	3	4
12	8	8	8	8	8	1	3
12	8	8	8	8	8	1	3
12	8	8	8	8	7	1	3
12	8	8	8	8	7	1	3
12	8	8	8	8	7	1	3
12	8	8	8	7	7	1	3

8 → $\frac{26}{64}$ 12 → $\frac{7}{64}$ 4 → $\frac{6}{64}$ 7 → $\frac{10}{64}$ 5 → $\frac{1}{64}$ 1 → $\frac{6}{64}$ 3 → $\frac{9}{64}$



0
110
101
100
1110
11111
11110

30. ZI 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.

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 = FBD1cx
RJ.: FBD1 FFFF
Dobro Šta nije 2+8 = A? Otkud B? Pise da nema carryja
je, ide FA D1 FF FF,
kak je ovo iznad dobro?? zar se carry ne prenosi samo izmedu parova 16 bitova? Mislim da je greška trebalo bi biti FBD1 FFFF jer se carry prenosi unutar 16 bitnih

iz multeha (http://www.fer2.net/showpost.php?p=1796573&postcount=19 - tema multimedijske tehnologije, znam da sam dobio sve bodove tu : sa zasićenjem : RIP fer2 :(

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 nečiji 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).

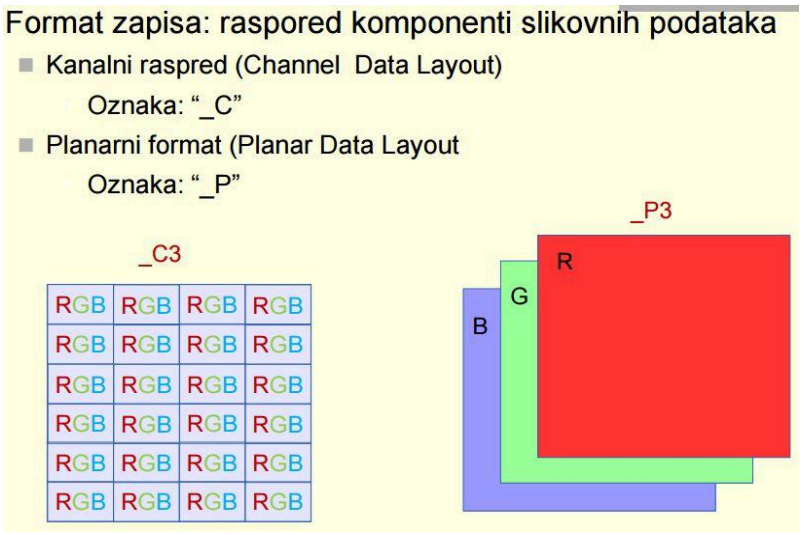
AKO ne prenosimo carry, rj. sa wraparoundom je 2221 7221? - ne, vec je 1 2221 7221 zbog objašnjenja ispod

moze neko objasnit ovaj wrap around? jel to normalno zbrajamo, ali onda maknemo preljev umjesto da zaokruzimo na FFFF? - tako je

Jel moramo dodati ovu jedinicu onda na pocetku? *1* 2221 7221 ili je mozemo maknut?
Ne mozes ju maknut. Ta jedinica nije preljev nego je bila tu od pocetka zadatka. Po meni je to kao da su zadali 0001 2345 6789 + 0000 FEDC 0A98, dakle ta jedinica nije preljev da ju mozes maknuti nego je konkretan 16 bitni broj.

Jel u wrap around se može prenijeti 1 iz prošlog 16 bitnog na idući npr da smo imali prijenos 1 iz ovog 7221 na ovaj u 2221 pa bi bilo 2222? **Ne. Vec je objasnjeno, ali basically to nije 1 broj od 32 bita nego skup brojeva od po 16 bitova. Znaci u zadacima ih tretiras kao odvojene brojeve i u slucaju preljeva imas 2 opcije - ako je sa zasicenjem onda promatranih 16 zaokruzis na FFFF, ako je wrap around onda odbacis preljev. Znaci konkretno za tvoj slucaj (wrap around), ako imas 1 iz proslog 16-bitnog onda to ne prenosis (jer ovaj iduci 16 bitni segment nema nikakve veze s trenutnim) nego zanemaris.**

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



- **Planarni raspored** podataka je da prvo imaš cijelo polje crvenih vrijednosti, pa cijelo polje zelenih vrijednosti, pa cijelo polje plavih
  i. For planar format, there is one value per pixel but potentially several related planes.
  RRRRRRRR
  RRRRRRRR
  RRRRRRRR
  RRRRRRRR
  GGGGGGGG
  GGGGGGGG
  GGGGGGGG
  GGGGGGGG
  BBBBBBBB
  BBBBBBBB
  BBBBBBBB
  BBBBBBBB
- **Kanalni raspored** je standardno, jedno polje sa RGB vrijednostima
  i. In channel format, all values share the same buffer and all values for the same pixel position are interleaved together.
  RGBRGBRGBRGBRGBRGBRGBRGB
  RGBRGBRGBRGBRGBRGBRGBRGB
  RGBRGBRGBRGBRGBRGBRGBRGB
  RGBRGBRGBRGBRGBRGBRGBRGB

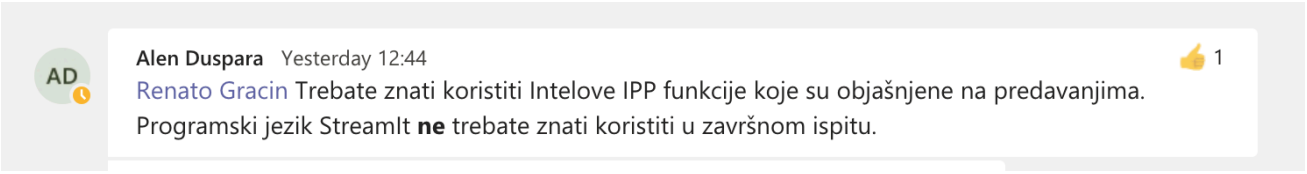
32. **ZI** Trebalo je dohvacati sliku s kamere koja vraca sliku u formatu **RGB 5:6:5**, a trebalo je koristiti funkcije **CAMERA_VSYNC()** koja vraca 1 ako je VSync high, inace 0, **CAMERA_HREF()** vraca 1 ako je HREF 1, slicno i za **CAMERA_CLOCK()**. Dakle trebalo je primiti njene podatke i to spremiti u normalnom RGB formatu u neki niz.
    - **RGB 5:6:5**
        i. 5 bitova za crvenu, 6 bitova za zelenu i 5 bitova za plavu

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

```
z=0;
   while(CAMERA_isVSYNdown(READ_CAM));
   while(CAMERA_isVSYNup(READ_CAM));


        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");
```

ZADACI S KODOVIMA NE ULAZE U ZI 2020/2021? MOŽE NETKO POTVRDITI?
S obzirom da će biti na zaokruživanje, sumnjam da će biti kodovi

NB - treba poznavati IPP funkcije, s Teamsa:



33. ZI 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;        //casta stream slike u neki njihov tip podatka

        Ipp8u ipp_destination[3][width*height] = {0};    // 3 1D polja za planarni prikaz

        IppiSize roiSize = {width, height};      //neka struktura kao arg funkcije

        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;
}
```

```
        /*  pSrc
```
Pointer to the source image ROI for pixel-order data. An array of pointers to the source image ROI in separate color planes in case of planar data.
```
        srcStep
```
Distance in bytes between starts of consecutive lines in the source image.
```
        pDst
```
Pointer to the destination ROI for pixel-order data. An array of pointers to destination buffers in separate color planes in case of planar data.
```
        dstStep
```
Distance in bytes between starts of consecutive lines in the destination image.
```
        roiSize
```
Size of the source and destination ROI in pixels.
```
         */
```

34. ZI Sto sve treba poslat iputem I2C sabirnice na video senzor OV5620 da bi on koristio 8 bitnu komunikaciju i da salje RGB komponente (redoslijed RGB komponenti nije bitan).

- Za pitanja ovakvog tipa pogledati registre senzora (slajd 42-44 zadnje prezentacije) i vidjeti o kojim se kontrolnim registrima radi.
- Npr, za RGB treba postaviti bitove 2 i 0 COM7 registra na 0 i 1. To znači da treba poslati 3 bajta: 1) adresu senzora, 2) adresu registra 0x12 i 3) te bitove koji su nam od interesa.
- Za 8-bit comms ne znam, čini mi se da se koristi po defaultu.

ZI 2021./2022.

(Tekst se odnosi na **1. i 2.** zadatak) Neka program sadrži dio O1 i O2 koji imaju udjele u vremenu izvođenja p1 = 35% i p2 = 60%. Dio O1 može se ubrzati 6 puta, a O2 4 puta.

1. Koliko je ubrzanje u slučaju optimizacije dijela O1?
   a. 1.41
   b. 1.50
   c. 1.82
   d. 1.95
   e. 2.03
   f. Ništa od navedenog

2. Koliko je ukupno ubrzanje u slučaju optimizacije oba dijela?
   a. 3.23
   b. 3.87
   c. 4.53
   d. 4.61
   e. 4.80
   f. Ništa od navedenog

(Tekst se odnosi na **3. i 4.** zadatak) Algoritam obrade podataka sastoji se od početnog dijela koji učitava blok od 50000 podataka čije izvođenje (učitavanje) traje 1000 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 2500 ns. Prilikom odgovaranja zaokružite rezultat na prvi veći cijeli broj.

3. Koliko je potrebno procesora da bi se algoritam ubrzao 4 puta?
   a. 3
   b. 4
   c. 5
   d. 6
   e. 7
   f. Ništa od navedenog

4. Kolika je paralelna efikasnost ako sustav ima 9 procesora?
   a. 77%
   b. 81%
   c. 75%
   d. 84%
   e. 79%
   f. Ništa od navedenog

U sljedećim zadacima **(5. – 11.)** razmatramo tekući i referentni blok, zadan je algoritam pretraživanja **ORT** (*Orthogonal Search*) i **MAD** kao mjera poremećaja. Pretpostavite da algoritam sliku **proširuje nulama (0)** ako je to potrebno (osim u slučaju *Full Search* algoritma). Koristi se Kartezijev koordinatni sustav (pozitivan smjer Y osi prema gore), a svaki korak pretraživanja računa se samo jedanput.

Tekući blok:

1	5	2	21	4	3	15	4	4	4
1	5	2	21	4	3	15	4	4	4
1	1	3	2	10	1	17	6	6	4
1	1	4	19	8	8	1	19	9	4
1	2	4	1	20	20	17	3	9	4
1	4	4	4	1	20	20	1	24	4
1	5	0	2	19	19	0	1	15	4
1	4	1	1	3	8	0	1	10	4
1	4	1	1	3	8	0	1	1	4
1	4	1	1	3	8	0	1	1	4

Referentni blok:

1	5	2	21	4	3	15	4	4	4
1	5	20	20	4	3	20	20	20	4
1	1	12	20	10	1	17	19	7	4
1	1	4	10	8	8	1	1	9	4
1	2	4	1	19	18	17	3	9	4
1	4	4	4	17	2	1	4	24	4
1	5	0	2	1	1	0	1	15	3
1	4	1	1	3	8	0	1	10	3
1	4	1	1	3	8	0	1	10	3
1	4	1	1	3	8	0	1	1	4

5. Koji je konačni vektor pomaka?
   a. (3, -3)
   b. (3, 3)
   c. (2, -3)
   d. (0, -2)
   e. (2, 3)
   f. Ništa od navedenog

6. Kolika je mjera poremećaja za konačni slikovni element na koji pokazuje vektor pomaka?
   a. 5.75
   b. 4.25
   c. 6.25
   d. 7.75
   e. 9.25
   f. Ništa od navedenog

7. Koliki je broj pretraživanja?
    a. 9
    b. 10
    c. 11
    d. 12
    e. 13
    f. Ništa od navedenog

8. Koliko se ubrzanje postigne u odnosu na algoritam potpunog pretraživanja (*Full Search*)?
    a. 6.23
    b. 6.75
    c. 7.69
    d. 9.00
    e. 11.11
    f. Ništa od navedenog

9. Kolika je mjera poremećaja za donji slikovni element (blok) u vertikalnoj fazi prvog prolaza algoritma?
    a. 9.75
    b. 10.75
    c. 5.75
    d. 14.75
    e. 17.75
    f. Ništa od navedenog

10. Kolika je mjera poremećaja za desni slikovni element (blok) u horizontalnoj fazi drugog prolaza algoritma?
    a. 6.25
    b. 8.75
    c. 9.00
    d. 9.50
    e. 16.50
    f. Ništa od navedenog

11. Kolika je mjera poremećaja za lijevi slikovni element (blok) u horizontalnoj fazi prvog prolaza algoritma?
    a. 13.50
    b. 13.25
    c. 12.00
    d. 4.25
    e. 9.25
    f. Ništa od navedenog

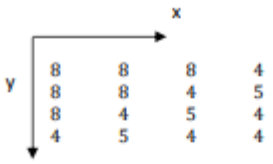
U slijedeća dva zadatka **(12. i 13.)** razmatramo operaciju zbrajanja korištenjem 16-bitne aritmetike sa zasićenjem:

```
  0 1 F F 5 4 7 8
+ F D D A A B 8 8
```

12. Koji je rezultat u slučaju korištenja zasićenja (*saturate*)?
    a. FFDA FFFF
    b. FFD8 0000
    c. FFFF FFFF
    d. FFD9 0000
    e. FFD9 FFFF
    f. Ništa od navedenog

13. Koji je rezultat u slučaju korištenja premotavanja (*wrap around*)?
    a. FFDA FFFF
    b. FFD8 0000
    c. 0000 0000
    d. FFD9 0000
    e. FFD9 FFFF
    f. Ništa od navedenog

U slijedeća četiri zadatka **(14., 15., 16. i 17.)** razmatramo zadani blok slikovnih elemenata P(x,y).



Blok je potrebno kodirati po sljedećem modelu:

Predviđanje slikovnog elementa P u rasponu od 0 do 255:

$$\hat{P}(x,y) = \text{int}\left(\frac{P(x-1,y)+P(x-1,y-1)+P(x,y-1)}{3}\right)$$

Ako je rezultat prethodne operacije < 0, onda je $\hat{P}$ =0, a ako je rezultat > 255, onda je $\hat{P}$ = 255. Predviđanje za slikovne elemente na rubovima gdje nisu definirani pojedini ulazni slikovni elementi uzima se uz pretpostavku da je njihova vrijednost 0.

Greška predviđanja: $e(x,y) = |P(x,y) - \hat{P}(x,y)|$

Greška predviđanja se kodira Huffmanovim kodovima generiranih pomoću binarnog stabla.

14. Kolika je prosječna duljina generiranih Huffmanovih kodova?
    a. 2,6
    b. 2,4
    c. 3
    d. 3,17
    e. 2,8
    f. Ništa od navedenog

15. Kolika je teorijska entropija izvora primijenjenog modela?
    a. ≅2,022
    b. ≅1,647
    c. ≅-2,022
    d. ≅-1,647
    e. ≅13,193
    f. Ništa od navedenog

16. Koliko je ukupno potrebno bitova za kodiranje zadanog bloka pomoću opisanog modela?
    a. 32
    b. 33
    c. 34
    d. 35
    e. 36
    f. Ništa od navedenog

17. Koliki je postignuti stupanj kompresije ako su slikovni elementi veličine 1 bajt?
    a. ≅3,55
    b. ≅3,66
    c. ≅3,77
    d. ≅3,88
    e. 4
    f. Ništa od navedenog

18. Za pretvorbu prostora boja koristi se funkcija *ippiRGBToYUV*. Izvorišna slika je polje 8-bitnih vrijednosti RGB komponenti zapisanih u slikovnom redoslijedu (pixel-order). Komponente Y,U i V odredišne slike su zapisane u planarnom zapisu. Koji format zapisa i tip podataka je potrebno koristiti u ovom slučaju?
    a. 8u_P3R
    b. 8u_C3R
    c. 8u_C3P3R
    d. 8u_AC4R
    e. 8u_AC4P4R
    f. Ništa od navedenog

Točni odgovori (provjereno): 1. A, 2. B, 3. C, 4. E, 5. E, 6. B, 7. A, 8. D, 9. D, 10. A, 11. A, 12. E, 13. D, 14. E, 15. A, 16. B, 17. D, 18. C

PROGRAMSKI ZADACI (Na kraju dokumenta su jer ne ulaze u MI/ZI ak godine 2020/2021)

Kaj jel bude to u 2021/2022 ili

Zadatak 1 (9 *bodova*): Napisati funkciju `void dct_quant_frame(int* data, int M, int N, int *qtable)` u programskom jeziku *C* koja nad blokom (okvirom) *MxN* cijelih brojeva provodi DCT transformaciju i kvantizaciju (kvantizacijski faktori se prenose pomoću pokazivača na polje od 64 elementa `*qtable`). Operacija se izvodi po blokovima veličine 8x8. DCT transformaciju bloka izračunati pomoću funkcije `dct_block` (napisati i ovu funkciju) nad blokom dimenzija 8x8 čiji prototip je `void dct_block(float *b)`. Pokazivač *b* se koristi za prijenos bloka u funkciju i za povratak vrijednosti. Prilikom pretvorbe iz `float` u `int` potrebno je izvršiti zaokruživanje na prvi najbliži cijeli broj (iskoristiti funkciju `long lroundf (float x)` iz `math.h`).
Napomena: dimenzije okvira *M* i *N* su proizvoljni brojevi koji ne moraju biti višekratnici broja 8, stoga je ulazni blok podataka potrebno proširiti sa nulama da bi ulazni blok dobio dimenzije koje su višekratnik broja 8.

```
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(int m = 0; m < M; m = m+8)
                for(int 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];
                        F_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;
}
```

35. ZI 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.
    - **RGB 5:6:5**
        i. 5 bitova za crvenu, 6 bitova za zelenu i 5 bitova za plavu

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

```
z=0;
   while(CAMERA_isVSYNdown(READ_CAM));
   while(CAMERA_isVSYNup(READ_CAM));


        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");
```

36. ZI **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 (I2C
Write(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)

Q: O čemu se ovdje radi? Tko je osoba na slici iznad?

A: Radi se o Gospodaru Sminemu, čuvaru svih bića koje pišu ispit iz MAIS-a.

Q: Aha! Ali, zašto se njegova slika nalazi na samom kraju ovog dokumenta?

A: Veliki Sminem pobrinut će se da svi, koliko god malo znali, prođu ovaj predmet.

Q: Hmm, ali što ako ja obrišem sliku Sminema?

A: Onaj koji obriše sliku ne samo da oskvrnjuje baštinu prijašnjih generacija koje su odabrali Sminema kao svog zaštitnika, nego mu i slijedi punih sedam godina propalih investicija.

„We're all gonna make it brah.” - David A. Huffman a.k.a. Daddy ZyZZ

Double Click on me to open lol.