
Skripta iz OAR

Teorija i zadaci

Nikola Vlahović

Teorijska pitanja

I Uvod

1.1 Koja je, uopšteno, razlika između organizacije i arhitekture računara?

Arhitektura računara je apstraktni model i programerov pogled u vidu instrukcija, načina adresiranja i registara tj. predstavlja attribute sistema koji imaju direktan uticaj na izvršenje programa. Računarska organizacija predstavlja realizaciju arhitekture. Arhitektura opisuje šta računar radi a organizacija kako on to radi.

A – Da li će računar imati instrukciju za množenje?

O – Kako će se ta instrukcija implementirati?

1.2 Koja je, uopšteno, razlika između structure i funkcije računara?

Struktura predstavlja način na koji su komponente u međusobnom odnosu dok funkcija predstavlja neku aktivnost tj. rad svake komponente kao dela strukture.

1.3 Koje su četiri glavne funkcije računara?

Obrada, skladištenje, premeštanje podataka i upravljanje (nad ovim 3 funkcijama).

1.4 Nabrojite i ukratko definišite glavne strukturne komponente računara.

CPU – Upravlja radom računara i izvodi njegove funkcije obrade podataka.

Glavna memorija – Skladišti podatke.

Sistemska međupovezivanje – Neki mehanizam koji omogućava komunikaciju između CPU, glavne memorije i U/I. Primer – sistemska magistrala.

1.5 Nabrojite i ukratko definišite glavne strukturne komponente procesora.

Upravljačka jedinica – Upravlja radom CPU-a a samim tim i celim računarom.

ALU jedinica – Izvodi funkcije obrade podataka računara.

Registri – Obezbeđuju unutrašnje skladištenje za CPU.

Međupovezivanje CPU – Neki mehanizam koji omogućava komunikaciju između gore 3 navedena.

II Evolucija i performansa računara

2.3 Na nivou integrisanog kola, koja su tri glavna sastavna dela računarskog sistema?

Logička kola, memorijske ćelije i međusobna veza tih elemenata.

2.6 Koje je ključno svojstvo kojim se odlikuje mikroprocesor?

Sve komponente procesora se nalaze na jednom čipu.

III Pogled odozgo na računarsku funkciju i međusobno povezivanje

3.2 Napravite listu i ukratko definišite moguća stanja koja definišu izvršenje instrukcije

1. Proračun adrese instrukcije – Određuje se adresa sledeće instrukcije za izvršenje.
2. Donošenje instrukcije – Instrukcija se učitava iz njene memorijske lokacije u procesor.
3. Dekodovanje operacije instrukcije – Analiza instrukcije zbog određivanja operacije za izvršenje i operandi koji treba da se koriste.
4. Proračun adrese operandi – Ako operacija obuhvata pozivanje na operand iz memorije ili U/I.
5. Donošenje operandi – Operand se donosi iz memorije ili U/I.
6. Operacija nad podacima – Izvodi se operacija iz instrukcije.
7. Skladištenje operandi – Upisuje se rezultat u memoriji ili U/I.

3.3 Napravite listu i ukratko definišite dva pristupa bavljenja višestrukim prekidima.

1. Dok se jedan prekid obrađuje, drugi se onemogućuje. Procesor ignoriše signal za prekid drugog prekida, on ostaje nerešen, a procesor će ga proveriti kada opet omogući prekide kad završi sa prvim prk.
2. Za prekide se definišu prioriteti i prekidi višeg prioriteta prekidaju rad upravljačkog programa za signal prekida nižeg prioriteta.

3.4 Koje vrste prenosa mora da podržava struktura za međusobno povezivanje u računaru tj. magistrala? Memorijska ka CPU, CPU ka memoriji, U/I ka CPU, CPU ka U/I, U/I ka memoriji ili od nje.

IV Keš memorija

4.1 Koje su razlike između sekvencijalnog pristupa, direktnog, i slučajnog pristupa?

Sekvencijalni pristup

U memoriji su grupe podataka organizovane u zapise koji slede jedan za drugim – sekvencijalni niz. Da bi se pronašao odgovarajući zapis mehanizam za čitanje/upisivanje mora da se premešta preko svih zapisa koji prethode traženom, pa se vreme pristupa različitim zapisima značajno razlikuje.

Direktan pristup

Zapisi imaju jedinstvenu adresu zasnovanu na njihovoj fizičkoj lokaciji. Pristup se ostvaruje direktnim pristupom. Vreme pristupa je takođe promenljivo.

Slučajan pristup

Svaki zapis ima jedinstven fizički ožičen mehanizam za adresiranje. Vreme pristupa je konstanto za sve zapise, nezavisno od njihove lokacije.

4.2 Kakav je opšti odnos između vremena pristupa, cene memorije i kapaciteta?

- Kraće vreme pristupa, veća cena po bitu.
- Veći kapacitet, manja cena po bitu.
- Veći kapacitet, duže vreme pristupa.

4.3 Kako se princip lokalnosti odnosi prema upotrebi višestrukih nivoa memorije?

Tokom izvršenja programa, memorijske reference za instrukcije i podatke teže da se grupišu. Tokom dugog vremena, grupe u upotrebi se menjaju ali u kratkom vremenu procesor uglavnom radi sa fiksiranim grupama reference memorije. Podaci se organizuju u hijerarhiju tako da procenat pristupa svakom sledećem nivou bude manji od onog prema nivou iznad. Vremenom neke grupe menjaju nivoe.

4.4 Koje su razlike između direktnog preslikavanja, asocijativnog, i asocijativnog preslikavanja skupa?

Direktno preslikavanje – Najjednostavnija tehnika, preslikava svaki blok gl. mem. u samo 1 mogući red keša.

Asocijativno preslikavanje – Dozvoljava svakom memorijskom bloku da se učita u bilo koji red keša.

-II- Skupa – Kompromis gornja 2, keš je podeljen na v skupova od kojih se svaki sastoji od k redova.

Direktno : $i = j \text{ modulo } m$; $i = \text{trenutni red keša}$, $j = \text{mem. blok}$, $m = \text{ukupan broj redova keša}$.

Asocijativno preslikavanje skupa : $m = v \times k$; $i = j \text{ modulo } m$

4.8 Koja je razlika između prostorne lokalnosti i vremenske lokalnosti?

Vremenska lokalnost : program veći broj puta pristupa jednoj istoj lokaciji za kratko vreme.

Prostorna lokalnost : program pristupa susednim memorijskim lokacijama.

Korišćenjem ova dva svojstva programa, umnogome se povećava broj keš pogodaka.

4.9 Uopšte, koje su strategije za iskorišćavanje prostorne lokalnosti i vremenske lokalnosti?

Vremenska lokalnost podiže nivo podataka i vrednosti u kešu nedavno korišćenih instrukcija.

Prostorna lokalnost koristi veće blokove keša i ima mehanizam za predonošenje u keš.

X Računarska aritmetika

10.1 Ukratko objasnite sledeća predstavljanja: označenu apsolutnu vrednost, komplement dvojke i polarizovano.

OAV – Najznačajniji bit u reči je bit predznaka (0 je +, 1 je -). U n-bit reči, n-1 bitova kranje desno drže apsolutnu vrednost celog broja. Neoznačen broj postaje označen dodavanjem bita predznaka.

Kom. 2 – Predstavlja operaciju promene znaka, gde se u broju svi bitovi nule zamenjuju sa bitovima jedinice i obrnuto (tako se dobija prvi komplement), pa se ta inverzija sabira sa 1.

Polarizovano – Invertovane su vrednosti bita predznaka i binarne vrednosti se oduzimaju sa 1.

pr. $+6 = 1101$, što bi kod OAV bilo -5. Znači + smo obeležili bitom 1 i broj 6 smo predstavili kao 5.

10.3 Koje je pravilo proširenja predznaka za brojeve u komplementu dvojke?

Bit predznaka se pomera na krajnju levu poziciju i sve se popunjava kopijama bita predznaka.

pr. 11101110 od 8 bita proširujemo na **11111111**1101110 od 16 bita jer je bit predznaka 1 (-).

10.4 Kako možete da formirate negaciju celog broja u predstavljanju pomoću komplementa dvojke?

Operacija negacije celog decimalnog broja se vrši tako što se napravi komplement svakog bita celog broja (uključujući i bit predznaka) i tako dobijenom broju se doda jedinica.

10.8 Koja su četiri suštinska elementa broja u notaciji u pokretnom zarezu?

Znak – 1 bit

Eksponent – 8 bitova

Mantisa – 24 bita

Vrednost broja : (Znak)Mantisa * $2^{\text{eksponent}}$

XII Skupovi instrukcija – karakteristike i funkcije

12.1 Koji su tipični elementi mašinske instrukcije?

Operacioni kod (OPKOD), referenca izvornog, referenca rezultujućeg operanda i referenca na sl. instrukciju.

12.2 Koje vrste lokacija mogu da drže operande izvora i odredišta?

Glavna ili virtuelna memorija, registar procesora, U/I uređaj.

12.3 Ako instrukcija sadrži četiri adrese, šta bi mogla biti namena svake od njih?

Dve služe za izvorne operande, jedna za odredišni operand (rezultat ide ovde) i jedna za adresu sledeće instrukcije.

12.4 Navedite i ukratko objasnite pet značajnih pitanja za projektovanje skupa instrukcija.

- Repertoar informacija : koliko i kojih operacija treba obezbediti i koliko trebaju biti složene.
- Tipovi podataka : razne vrste podataka nad kojima se izvode operacije.
- Format instrukcije : dužina instrukcije u bitovima, broj adresa, veličina polja itd.
- Registri : broj procesorskih registara koji mogu da se referenciraju instrukcijama i njihova upotreba.
- Adresiranje : načini na koje se specifikira adresa operanda.

12.5 Koje vrste operanada su tipične u skupovima mašinskih instrukcija?

- Registarski operandi -- koristi se vrednost iz registra koji je naveden kao operand.
 - Neposredni operandi -- koristi se vrednost konstante koja se navodi kao operand.
 - Memorijski operandi -- koristi se vrednost iz memorije na adresi koja se zadaje kao operand.
- Vrednosti mogu biti adrese, brojevi, znakovi, logički podaci.

12.11 Navedite tri moguća mesta za sladištenje povratne adrese za povratak iz procedure.

Registar, početak pozvane procedure, vrh steka.

XIII Skupovi instrukcija – načini i formati adresiranja

Pitanja su od 13.1 do 13.7 tj. definisanje svih načina adresiranja. Slika 13.1 je na strani 453.

P.1 Skicirati i objasniti kako radi neki od modova adresiranja (može da zapadne bilo koji).

Slika 13.1 , strana 453.

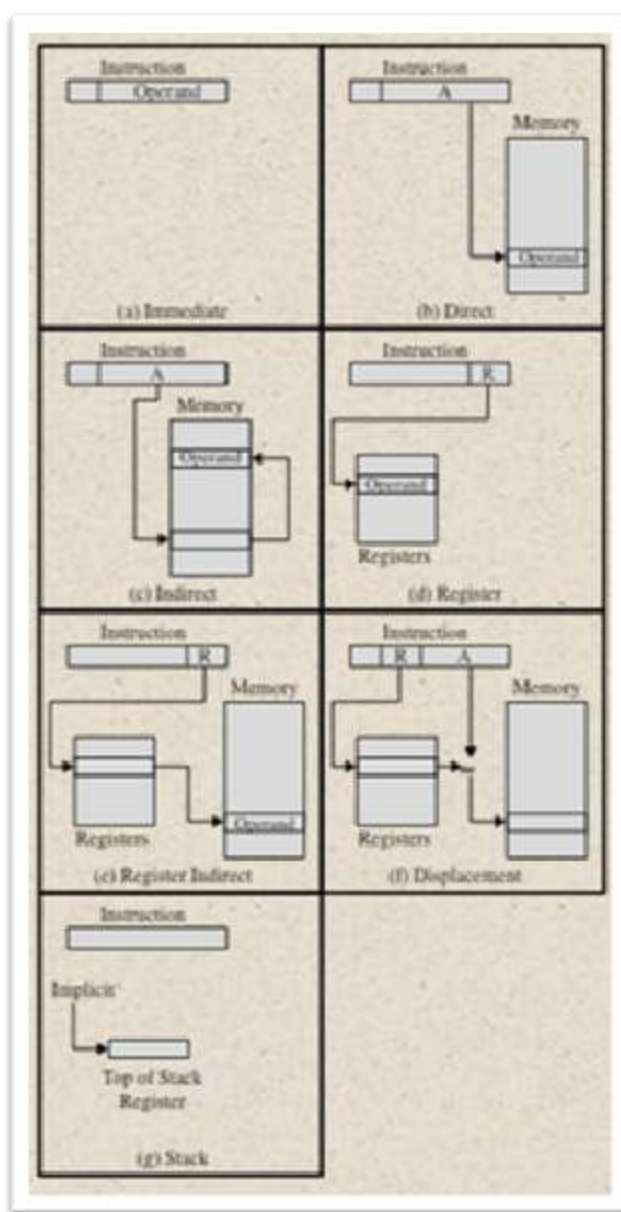
<i>Neposredno</i>	Operand = A	– Vrednost operanda je prisutna u instrukciji.
<i>Direktno</i>	EA = A	– Polje za adresu sadrži efektivnu adresu operanda.
<i>Indirektno</i>	EA = (A)	– U adresnom polju se referencira na adresu reči u memoriji koja sadrži adresu operanda potpune dužine.
<i>Registarsko</i>	EA = R	– Slično direktnom samo što adresno polje referencira registar.
<i>Registarsko ind.</i>	EA = (R)	– Isto kao indirektno samo što referencira registar.
<i>Sa pomerajem</i>	EA = A + (R)	– Zahteva da instrukcija ima 2 adresna polja. Vrednost koja se nalazi u jednom polju (A) koristi se direktno, a drugo se odnosi na registar čiji sadržaj se dodaje A za dobijanje ef. adrese.

Pomoću steka EA = Vrh steka – Stek je linearni niz lokacija. Stavke se dodaju na vrh steka. Njemu je pridružen pokazivač čija je vrednost adresa vrha steka. Pokazivač steka se čuva u registru pa prema tome reference na lokacije steka u memoriji su u stvari registarske indirektno adrese. Način adresiranja pomoću steka je oblik implicitnog adresiranja.

P.2 Osnovni načini adresiranja tabela (Mode, Algorithm, Advantage, Disadvantage).

- Ovo češće bude na ispitu.

Način	Algoritam	Prednost	Nedostatak
Neposredno	Operand = A	Nema reference memorije	Ograničena veličina operanda.
Direktno	EA = A	Jednostavno.	Ograničen adresni prostor.
Indirektno	EA = (A)	Veliki adresni prostor.	Višestruke reference memorije.
Registarsko	EA = R	Nema reference memorije.	Ograničen adresni prostor.
Registarsko ind.	EA = (R)	Veliki adresni prostor.	Dodatna memorijska referenca.
Sa pomerajem	EA = A + (R)	Fleksibilnost.	Složenost.
Pomoću steka	EA = Vrh steka	Nema reference memorije.	Ograničena primenljivost.



V Unutrašnja memorija

5.1 Koje su ključne osobine poluprovodničke memorije?

- Imaju 2 stabilna stanja koja se koriste za predstavljanje binarnih cifara 0 i 1.
- One su takve da se u njih upisuje da bi se uspostavilo stanje
- One su takve da se iz njih čita da bi se saznalo njihovo stanje.

5.2 Koje su dve interpretacije termina RAM memorija?

- Statička RAM (SRAM) memorija koja drži podatke u sebi bez potrebe sa njihovim obnavljanjem
 - Dinamička RAM (DRAM) memorija koja se prazni pa zahtevaju povremeno osvežavanje naelktrisanja.
- * Obe memorije gube podatke nestankom napajanja tj. isključivanjem računara.

5.3 Koja je razlika između DRAM i RAM memorije u smislu primene?

- SRAM je mnogo skuplji pa se koristi za keš, dok se DRAM koristi za sistemsku memoriju.

5.4 Koja je razlika između DRAM i RAM memorije u smislu karakteristika (brzina, veličina, cena)?

- SRAM je brža od DRAM, ali je DRAM mnogo jeftinija pa se ona koristi za veće memorije pa je i veće veličine jer je gušća (veći broj mem. ćelija na površini).

5.5 Objasnite zašto se jedna vrsta RAM memorije smatra analognom, a druga digitalnom.

- Analogne memorije rade kontinualno zavisno od njihovih ulaza (DRAM će biti u stanju 0 ili 1 u zavisnosti od prisustva ili odsustva napona na kon.), dok digitalne memorije rade sa logičkim elementima.

5.6 Koje su neke od primena ROM memorija?

- Mikroprogramiranje, primena kod sistemskih programa, tabela funkcija, biblioteka potprograma za često zahtevane funkcije.

5.7 Koje su razlike između EPROM, EEPROM i fleš memorija?

- **EPROM (Erasable PROM)** je ROM koji može biti obrisani i reprogramiran tako što se mali stakleni prozor na vrhu njegovog kućišta osvetli ultraljubičastom svetlošću 20ak minuta.
- **EEPROM (Electronically EPROM)** omogućava brisanje pod kontrolom softvera. U njega se podatak može upisati u bilo kom trenutku i bez brisanja prethodnog sadržaja jer se samo menjaju adresni bajtovi.
- **FLASH** je dobio ime zbog brzine kojom se može reprogramirati. Može se obrisati za svega nekoliko sekundi, mnogo brže od EEPROM-a. Danas se često koristi za smeštanje BIOS-a.

5.8 Objasnite funkciju svakog pina na slici 5.4b

Vcc – Napajanje električnom energijom za čip. ; Vss – Pin za uzemljenje

D1-D4 – Pinovi za podatke koji trebaju da se pročitaju

A0-A10 – Pinovi za adrese reči kojima se pristupa.

WE – Pin za omogućavanje upisa

OE – Pin za omogućavanje ispisa/čitavanja

RAS – Pin za izbor reda

CAS – Pin za izbor kolone

NC – Nepovezan (not connected) pin da bi broj pinova bio paran.

5.9 Šta je bit parnosti?

- Bit parnosti, ili bit prove, je bit koji je dodan na kraj binarnog koda koji pokazuje da li je broj bitova u stringu sa vrednošću jedan paran ili neparan. Parnost bita je najjednostavniji oblik koda detekcije greške.

5.10 U čemu se razlikuje SDRAM memorija od obične DRAM memorije?

- Za razliku od asinhronne DRAM memorije, SDRAM memorija razmenjuje podatke sa procesorom sinhronizovano u odnosu na signal generatora takta i radi na punoj brzini magistrale bez nametanja stanja čekanja.

VI Spoljašnja memorija

6.5 Definišite termine staza, cilindar i sektor.

- Osnovni elementi u organizaciji diska su **staze** koje su raspoređene kao koncentrični krugovi na pločama. One se dele na **sektore**, najmanje veličine prostora na koje mogu da se upišu podaci. **Cilindar** čine staze sa svih ploča koje su na istom poluprečniku.

6.6 Koja je veličina sektora tipičnog diska?

- 512 bajtova.

6.7 Definišite termine vreme pozicioniranja, rotaciono kašnjenje, vreme pristupa i vreme prenosa.

- Vreme pozicioniranja je vreme potrebno da bi se glava postavila na stazu.
- Rotaciono kašnjenje je vreme koje je potrebno da bi početak sektora stigao do glave.
- Vreme pristupa je zbir vremena pozicioniranja i rotacionog kašnjenja tj. to je vreme potrebno da se dođe na položaj za čitanje ili upisivanje.
- Vreme prenosa je vreme potrebno da glava izvede operaciju čitanja ili upisivanja nad sektorima.

6.8 Koje zajedničke karakteristike dele svi RAID nivoi?

- RAID je skup diskova koje operativni sistem vidi kao jedan logički disk.
- Podaci su raspodeljeni na fizičkim uređajima jednog niza u šemi poznatoj kao stratifikacija (striping).
- Kapacitet redundantnog diska koristi se za skladištenje informacija parnosti što nam garantuje mogućnost obnove podataka u slučaju otkaza nekog diska.

VII Ulaz/Izlaz

7.1 Navedite tri široke klasifikacije spoljašnjih, ili perifernih uređaja.

- Čitljivi za ljude, čitljivi za mašinu i komunikacioni uređaji.

7.2 Šta je međunarodna referentna azbuka?

- Najčešće korišćen tekstualni kod u kome se svaki znak predstavlja jedinstvenim 7-bitnim binarnim kodom tj. samim tim mogu se predstaviti 128 različita znaka.

7.3 Koje su glavne funkcije U/I modula?

- Upravljanje i vremensko usklađivanje, komunikacija procesora, komunikacija uređaja, baferovanje podataka i otkrivanje grešaka.

7.4 Navedite i ukratko definišite tri tehnike za izvršavanje U/I.

- Programirani U/I – Podaci se razmenjuju između CPU-a i U/I modula. Procesor izvršava program koji mu daje direktno upravljanje nad U/I operacijom. CPU mora da čeka dok se ne završi U/I operacija.
- U/I upravljan prekidima – Procesor izdaje U/I komandu i nastavlja da izvršava druge instrukcije, a U/I modul ga prekida kada završi svoj posao.
- Direktn pristup memoriji – U/I modul i glavna memorija razmenjuju podatke direktno, bez učešća CPU

7.5 Koja je razlika između memorijski preslikanog U/I i izolovanog U/I?

- Kod memorijskog preslikanog postoji jedan adresni prostor za mem. lokacije i U/I uređaje, dok kod izolovanog adresni prostor za U/I je izolovan od onog za memoriju.

7.6 Kada se pojavi prekid uređaja, kako procesor određuje koji uređaj je izdao prekid?

- Svaki uređaj ima jedinstven ID (adresu) te CPU upućuje U/I komande na odgovarajuću adresu uređaja.

7.7 Kada DMA modul preuzima upravljanje nad magistralom i dok on zadržava upravljanje nad magistralom, šta radi procesor?

- CPU je forisran da privremeno obustavi svoju operaciju, ovo se zove krađa ciklusa.

XIV Struktura i funkcija procesora**14.1 Koje opšte uloge imaju registri procesora?**

- Registri vidljivi za korisnike omogućavaju programeru da optimalnim korišćenjem registara svede referenciranje glavne memorije na najmanju moguću meru.
- Upravljačke i statusne registre upravljačka jedinica koristi za upravljanje radom procesora, a privilegovani korisnici za upravljanje izvršavanjem programa.

14.2 Koje kategorija podataka obično podržavaju registri vidljivi za korisnike?

- Registri opšte namene/podataka/adresa i uslovni kodovi.

14.3 Koja je funkcija uslovnih kodova?

- Predstavljaju bitove koje CPU dobija kao rezultat operacija. Osim samog rezultata operacije dobijamo i bitove koji nam govore da li je u operaciji bilo prenosa, da li je rezultat bio nule, da li je rezultat pozitivan ili negativan, ima li prekoračenja...

14.5 Šta je statusna reč programa?

- To je upravljački registar koji sadrži bitove stanja procesora . Sastoji se od akumulatora i flegova.

14.6 Navedite i ukratko objasnite razne načine na koje protočna obrada instrukcija može da se nosi sa instrukcijama uslovnog granjanja.

- Višestruki tokovi – Kopiranje inicijalnih delova protočne obrade što joj kroz korišćenje dva toka omogućava da donese obe instrukcije.
- Pretprihvatanje cilja granjanja – Čim se ustanovi uslovno granjanje, osim instrukcije koja sledi granjanje unapred se donosi i cilj granjanja.
- Bafer petlje – Bafer predstavlja malu, brzu memoriju koja sadrži najskorije donesene instrukcije. Ukoliko dođe do granjanja, hardver najpre proverava da li je njegov cilj već u baferu i ako jeste donosi sledeću instrukciju iz bafera.
- Predviđanje granjanja – Postoje razne tehnike za predviđanje, najčešće se koriste one koje predviđa da se nikad neće preduzeti, da će se uvek preduzeti i predviđanje na osnovu operacionog koda.
- Odloženo granjanje – Automatski se preraspoređuju instrukcije u okviru programa tako da se instrukcije granjanja pojave kasnije nego što želimo.

XI Digitalna logika

11.1 napravite tablicu istinitosti za sledeće Booleove izraze:

- $ABC + A'B'C'$
- $ABC + AB'C' + A'B'C'$
- $A(BC' + BC)$
- $(A + B)(A + C)(A' + B')$

Znak puta = I ; Znak plus = Ili

Kada piše ABC, to znači $A*B*C$ tj. A i B i C moraju biti jedan da bi rezultat bio jedan.

Kada piše $A'B$, to znači $A'*B$ tj. A' i B moraju biti jedan da bi rezultat bio jedan.

Kada piše $ABC + A'B$ to znači $A*B*C + A'*B$ tj. A i B i C moraju biti jedan ILI A' i B moraju biti jedan da bi rezultat bio jedan. Znači ili jedno od to dvoje mora biti jedan ili mogu i oba biti jedan da bi rezultat bio jedan.

U primeru a) Napravite tabelu za A, za B, za C, i za ABC. Pošto ima 3 slova tj. 3 bita, $2^3 = 8$ imate cifre od 0 do 7 (tj. 000 gde je A=0, B=0 i C=0 do 111 gde su svi 1). ABC će biti u ovom slučaju jedan samo kada su svi jedan a to je kod cifre 7. Isto tako u produzetku napravite tabelu za A', B' i C', oni će imati vrednosti suprotne od njihovih pravih znakova (tj. gde je A = 0, A' = 1), i isto tako napravite polje A'B'C' koje će biti jedan samo kada su sva tri inverzna jednaka 1 (u ovom slučaju kod cifre nula). Na kraju napravite jedno poslednje polje F tj. $ABC + A'B'C'$ koje će biti jedan kada je jedno od ova dva jedan ili kada su oba jedan. Pošto je $ABC = 1$ kod cifre 7 i $A'B'C' = 1$ kod cifre 0, ta dva polja će u ovoj koloni imati vrednost 1 a sva ostala polja će imati vrednost nula.

b) $A i B i C$ ILI $A i B i C'$ ILI $A' i B' i C'$ mora biti 1 da bi funkcija bila 1.

c) $ABC' * ABC' = ABC'$. Broj koji se množi samim sobom je jednak sebi, te je f-ja kod A=1, B=1 i C'=0 jedan.

A	B	C	ABC	A'	B'	C'	A'B'C'	F
0	0	0	0	1	1	1	1	1
0	0	1	0	1	1	0	0	0
0	1	0	0	1	0	1	0	0
0	1	1	0	1	0	0	0	0
1	0	0	0	0	1	1	0	0
1	0	1	0	0	1	0	0	0
1	1	0	0	0	0	1	0	0
1	1	1	1	0	0	0	0	1

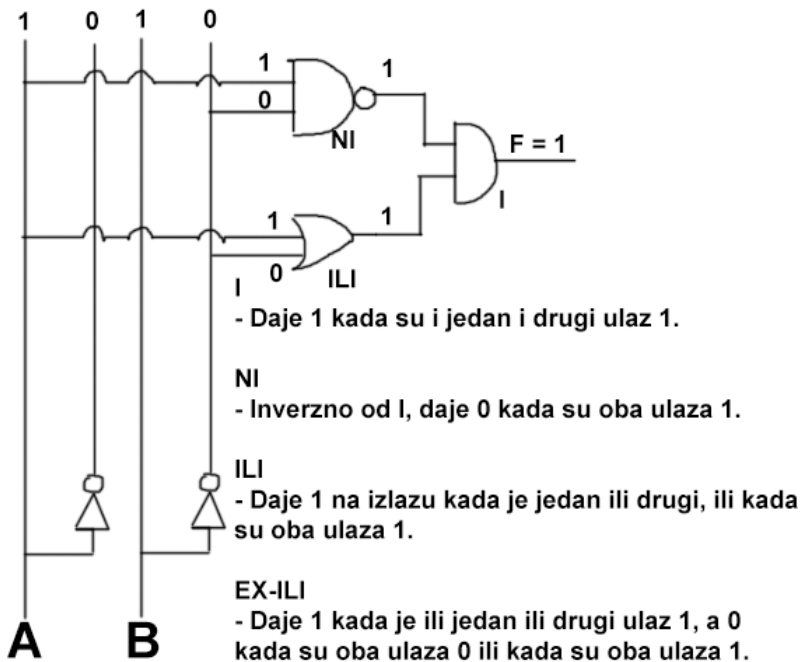
11.5 Konstruišite operaciju ekskluzivno ILI iz osnovnih Boleovih operacija I, ILI i NE.

- Ako imamo I i ILI kola, to znači da možemo da koristimo i NI i NILI kola (oni imaju inverzne vrednosti, crtaju se isto samo što im se dodaje kružić ispred). EX-ili kolo je 1 samo ako je jedan od dva ulaza jedan, što znači da ako su oba 1 ili oba 0 izlaz kola je 0.

- Postoji više načina za rad, ali suština je da nađemo kombinaciju funkcija koje na izlazu daju XOR = 1.

A	B	I	NI	ILI	NILI	XOR
0	0	0	1	0	1	0
0	1	0	1	1	0	1
1	0	0	1	1	0	1
1	1	1	0	1	0	0

- Primetimo da kada je $A = 1$ i $B = 0$ da su nam funkcije NI i ILI = 1, a uz njih je i traženi XOR = 1. Ukoliko bismo iskoristili I (AND) funkciju na NI i ILI kola gde su ona 1, dobili bismo rezultat 1.

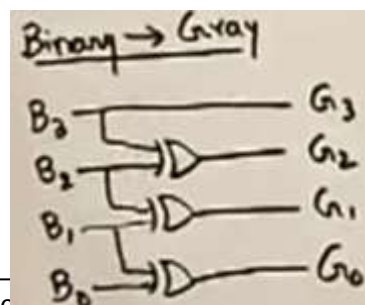


11.11 Projektujte kolo koje binarni kod pretvara u Grejov kod.

- Najvažniji bit (skroz levo) je uvek isti, potom se radi XOR f-ja sa tim bitom i bitom pored, a nakon toga se radi XOR sa bitom u sredini i skroz desnim bitom. Pogledajte klip ispod, na njemu je sve objašnjeno.

<https://youtu.be/iH6UWZoTl2c?t=110>

Dakle, uvek je prethodni ulaz prvi ulaz za XOR kolo ispod.



Operativni sistem

1. Šta je operativni sistem?

- Skup programa koji upravljaju hardverom, podacima i izvršavaju naredbe korisnika.

2. Navedite i ukratko definišiti ključne usluge koje pruža OS?

- Raspoređivanje procesa – određivanje koji će se proces izvršavati u bilo kom trenutku.

- Upravljanje memorijom – uključuje mogućnost i virtuelne memorije.

* Odg. iz PARIOSA: upravljanje procesorom, memorijom, I/O uređajima, podacima i aplikacijama.

3. Navedite i ukratko definišite glavne vrste raspoređivanja u OS?

- Dugoročno raspoređivanje – Odluka da se doda skupu procesa za izvršenje.

- Srednjovečno -| - - -II- procesa koji su delimično ili potpuno u glavnoj memoriji.

- Kratkoročno –II- - Odluka koji od raspoloživih procesora će se izvršiti pomoću procesora.

- U/I –II- – Odluka koji od nerešenih U/I zahteva procesa će se opslužiti pomoću datog U/I uređaja.

4. Koja je razlika između procesa i programa?

Program kada uđe u memoriju postaje proces i tada se izvršava. Proces je program koji se izvršava.

5. Koja je namena razmenjivanja?

- U/I je suviše spor u poređenju sa CPU toliko da on može da bude besposlen u najvećem delu vremena. Zato se procesi razmenjuju u memoriji, kako prostor na njoj postane raspoloživ. Kada se proces završi, pomera se napolje iz glavne memorije.

Z1 (3).pdf – Zadatak sa flegovima.

Zero Flag – Setovan kada je rezultat aritmetičko-logičke operacije 0.

Sign Flag – Setovan kada je najvažniji (poslednji (skroz desno)) bit 1.

Carry Flag – Setovan kada se pojavi prenos.

Overflow Flag – Setovan kada dodje do prekoračenja.

Auxiliary Flag – Setovan kada postoji prenos posle znaka.

1. Kako su postavljeni flegovi posle 8-bit HEX sabiranja?

D7h + CAh Carry **11 1111**
 D7h = 11010111 (2) 11010111
 CAh = 11001010 (2) + 11001010
 = **110100001** (Ignorišemo poslednji prenos)
 10100001 (2) = **A1h**

ZF = 0

SF = 1

CF = 1

OF = 0

AF = 1

2. Kako su postavljeni flegovi posle 8-bit HEX sabiranja?

38h + C8h Carry **11111**
 38h = 00111000 (2) 00111000
 C8h = 11001000 (2) + 11001000
 = **100000000** (Ignorišemo poslednji prenos)
 00000000 (2) = **00h**

ZF = 1

SF = 0

CF = 1

OF = 0

AF = 1

Z1 (8).pdf – Zadatak sa sabiranjem dvojnih komplementa.

Konvertovati dati par decimalnih brojeva A i B na 8-bitni dvojni komplement i izvršiti aritmetičku operaciju sabiranja. Rezultat R konvertovati u decimalni oblik. Pokazati da u slučaju prekoračenja dobijamo netačan rezultat.

$$74 = \underline{\hspace{2cm}} \quad 74 = 01001010_{(2)}, \text{ komplement } 10110110_{(2)}$$

$$58 = \underline{\hspace{2cm}} \quad 58 = 00111010_{(2)}, \text{ komplement } 11000110_{(2)}$$

Netačan rezultat : ----- + ----- = ----- $-74 + (-58) = 124$

Ispravan rezultat, ako imamo više bitova : ----- + ----- = ----- $-74 + (-58) = -132$

Prevođenje iz osnove 10 u 2.

0	0	0	0	0	0	0	0	0	0	0
1024	512	256	128	64	32	16	8	4	2	1
2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Pošto zadatak kaže da se 8 bita koristi, ukoliko iskoristimo manje bitova popunjavamo ostatak nulom sa leve strane.

Dvojni komplement dobijamo kada broju u binarnom obliku zamenimo sve nule sa jedinicama u sve jedinice sa nulama i saberemo to sa jedan. Njime dobijamo suprotan znak trenutnog broja.

$$\begin{array}{rcl}
 74 = 01001010 & & 1 \\
 \text{Komplement} - & 10110101 & \\
 + & 1 & \\
 = & 10110110 = -128+32+16+8+4+2 = -74 &
 \end{array}
 \quad
 \begin{array}{rcl}
 58 = 00111010 & & 1 \\
 \text{Komplement} - & 11000101 & \\
 + & 1 & \\
 = & 11000110 = -128+64+4+2 = -58 &
 \end{array}$$

$$\begin{array}{r}
 \textcolor{red}{1}0 \quad \textcolor{blue}{1}1 \\
 \textcolor{blue}{1}0110110 \\
 + \textcolor{blue}{1}1000110 \\
 \textcolor{red}{1}01111100 \text{ (Ignorišemo poslednji prenos)} = 64+32+16+8+4 = 124 - \text{Netačno}
 \end{array}$$

Kod sabiranja 2 broja različitog znaka nikada ne dolazi do prekoračenja. Do prekoračenja dolazi kada sabiranjem dva pozitivna broja dobijemo negativan broj (0+0 daju 1 skroz levo) ili dva negativna broja daju pozitivan broj (1+1 (plavi) daju 0 (zelena) (vidi primer iznad)). Overflow se rešava proširivanjem opsega, te u ovom slučaju ću proširiti na 10 bita. Znači dodajemo 2 nule sa leve strane brojevima, ali pošto se radi dvojni komplement te nule postaju jedinice.

$$\begin{array}{rcl}
 74 = 0001001010 & & 1 \\
 \text{Komplement} - & 1110110101 & \\
 + & 1 & \\
 = & 1110110110 & \\
 1110110110 & & \\
 + 1111000110 & & 2^{10} = 1024 \\
 \textcolor{red}{1}1101111100 \text{ (ignorišemo poslednji bit prenosa), } & -1024+512+256+64+32+16+8+4 = -132. &
 \end{array}
 \quad
 \begin{array}{rcl}
 58 = 0000111010 & & 1 \\
 \text{Komplement} - & 1111000101 & \\
 + & 1 & \\
 = & 1111000110 &
 \end{array}$$

1. Konvertovati dati par decimalnih brojeva na 8-bitni dvojni komplement i izvršiti naznačene aritmetičke operacije. Rezultat konvertovati u decimalni oblik. Pokazati da u slučaju prekoračenja dobijamo netačan rezultat.

A 74+58

74

37	0
18	1
9	0
4	1
2	0
1	0
0	1

58

29	0
14	1
7	0
3	1
1	1
0	1

c8c7

```
01111010
01001010
00111010
-----
10000100
```

Pozitivan + Pozitivan = Negativan ---- prekoračenje!

Osim toga c8 XOR c7 = 1

Rezultat : -128 + 4 = -124

Ispravan rezultat, ako imamo više bitova : 128 + 4 = 132

B 13 + (-36)

C (-54) + (-78)

2. Dati su brojevi kao 8-bitni dvojni komplementi. Izvršiti naznačene aritmetičke operacije i pokazati pojavu prekoračenja.

A 10011001 – 00011110

- 00011110
11100010 inverzija svih bitova i dodatak 1

10

```
10011001
+ 11100010
-----
01111011
```

Negativan + Negativan = Pozitivan ---- prekoračenje!

Osim toga c8 XOR c7 = 1

B 11110010 + 10010101

C 10110110 + 11101101



Brojeve $A=87$ i $B=-47$ predstavi u 8- bitnoj reprezentaciji dvojnog komplementa.

$$A=87=0101\ 0111$$

$$B=-47=1101\ 0001$$

Izračunaj $A+B$ u dvojnog komplementu.

$$\begin{array}{r} 0101\ 0111 \\ 1101\ 0001 \\ \hline 10010\ 1000 \end{array}$$

$$A+B = 0010\ 1000 = 40.$$

Izračunaj $A-B$ u dvojnog komplementu.

Da li ima pprekoračenja (overflow)? Objasni.

$$A-B = A + (-B) = 0101\ 0111 + 0010\ 1111$$

$$\begin{array}{r} 0101\ 0111 \\ 0010\ 1111 \\ \hline 1000\ 0110 \end{array}$$

$$A-B = 1000\ 0110 = -122$$

Overflow! Bitovi znaka su različiti.



Brojeve $A=87$ i $B=-47$ predstavi u 12- bitnoj reprezentaciji dvojnog komplementa.

Izračunaj $A-B$ u dvojnog komplementu.

Da li ima pprekoračenja (overflow)? Objasni.

$$A=87=0000\ 0101\ 0111$$

$$B=-47=1111\ 1101\ 0001$$

$$-B = 0000\ 0010\ 1110 + 1 = 0000\ 0010\ 1111$$

$$A-B = A+(-B) = 0000\ 0101\ 0111 + 0000\ 0010\ 1111 = 0000\ 1000\ 0110 = 134$$

Nema prekoračenja!

Z1 (6).pdf – Zadatak sa analizom x8086 programa.

Pomnožiti AX sa 26, koristeći "shifting" i "addition" instrukcije (napomena: $26 = 16 + 8 + 2 = 2^4 + 2^3 + 2^1$).

Analiziraj sekvencu i odgovori na postavljena pitanja:

```

mov ax,02h ; test value
mov dx,ax ; DX = ____?      DX = AX = 02h = 00000010b
shl dx,4 ; DX = ____?      DX = 00100000b          ; AX * 16
push dx ; _____      Vrednost DX je kopirana u stek. ; save for later
mov dx,ax ; DX = ____?      DX = AX = 02h = 00000010b
shl dx,3 ; DX = ____?      DX = 00010000b          ; AX * 8
shl ax,1 ; AX = ____?      AX = 00000100b          ; AX * 2
add ax,dx ; AX = ____?      AX = 00010100b = 14h      ; AX * 10
pop dx ; DX = ____?      DX = 00100000b          ; recall AX * 16
add ax,dx ; AX = ____?      AX = 00110100b = 34h      ; AX * 26

```

AX (uz BX, CX i DX) je registar opšte namene koji su veličine 16 bita ali mogu se koristiti kao i dva odvojena 8-bitna registra. Viši bajt (bajt = 8 bita) AX se naziva AH a niži bajt se naziva AL. Većina instrukcija dozvoljava korišćenje ovih 8-bitnih registara kao operande.

MOV – Kopira podatke sa jedne lokacije na drugu tj. kopira operand2 u operand 1.

Primeri : operand 1 = operand 2

mov AX, 02h – Kopira 02h (00000010) u AX tj. AX = 02h.

mov DS, AX – Kopira vrednost AX-a u DS.

mov CL, 'A' – CL = 41h (u ASCII kodu simbol A ima kod 41h)

SHL – Šiftuje operand 1 ulevo. Broj pomeraja određuje operand 2. Bit koji ispada ide u CF (Carry Flag) a sa desne strane nadolaze nule.

Primer:

MOV AL, 11100000b ; AL = 11100000

SHL AL, 1 ; AL = 11000000, CF = 1

SHR – Šiftuje operand 1 udesno. Broj pomeraja određuje operand 2. Bit koji ispada ide u CF (Carry Flag) a sa leve strane nadolaze nule.

Primer:

MOV AL, 11100000b ; AL = 11100000

SHR AL, 1 ; AL = 01110000, CF = 0

PUSH – Ubacuje vrednost u stek i stavlja je prvo neiskorisceno mesto pri vrhu.

POP – Uzima prvu vrednost iz steka na koju naleti.

Primer:

MOV CX, 3 ; CX = 3 = 0000 0011

PUSH CX ; U prvo slobodno mesto pri vrhu steka upisuje se 0000 0011

POP AX ; Uzima se prvo upisano mesto pri vrhu i ubacuje u AX, te AX je sada jednako 3.

ADD – Sabira.

Primer : operand1 = operand1 + operand2

MOV AL, 5 ; AL = 5

ADD AL, -3 ; AL = 2

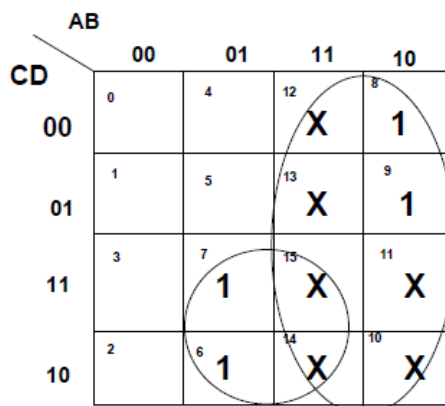
		CX = 0000 0011	
		PUSH CX:	POP AX:
		0000 0000	AX = 0000 0011
		0000 0011	
		0110 1100	
		1101 0110	
Stek			
0000 0000	<- Prazan		
0000 0000	<- Prazan		
0110 1100			
1101 0110			

Primeri zadataka vezanih za projektovanje logičkih kola

1. Projektujte logičko kolo koje proizvodi 1 na izlazu kada je unos veći ili jednak od 6.

- U ovim zadacima sve što je veće od devet obeležavamo sa X i možemo a i ne moramo da ih koristimo.
- Možete da koristite i tabelu sa prve strane (pošto na ovoj 0,1,3,2 idu vertikalno a ne horizontalno).

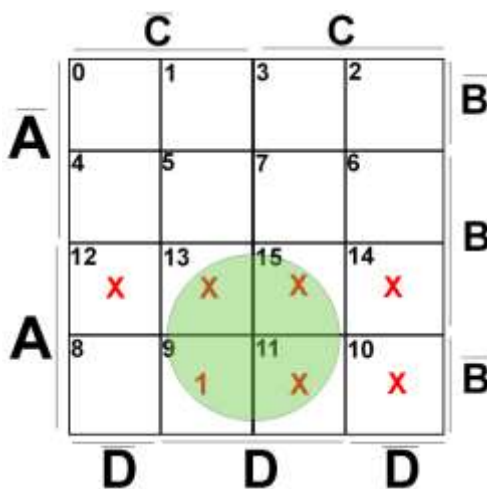
Inputs				Output
A	B	C	D	X
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X



$$f = (A,B,C,D) = A + BC$$

2. Projektovati logički sklop koji otkriva pojavu broja 9 zapisanog u BCD kodu.

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X



$$Y = AD$$

Koder

- Predstavlja kombinacionu mrežu sa više ulaza (**m**) i više izlaza (**n**) koja obavlja funkciju kodovanja informacija. Informacija je signal koji je doveden na samo **jedan** od ulaza dok se na izlazu dobija kodovana informacija u obliku binarnog broja sa **n** cifara (npr. ako je signal na trećem ulazu onda se na izlazu dobija 010 (binarno 2, dva jer brojanje kreće od nule). Dakle, samo 1 ulaz može dovesti signal, ako 2 ili više ulaza istovremeno dovedu signal na ulaz koder, na izlazu će se generisati pogrešan kod.

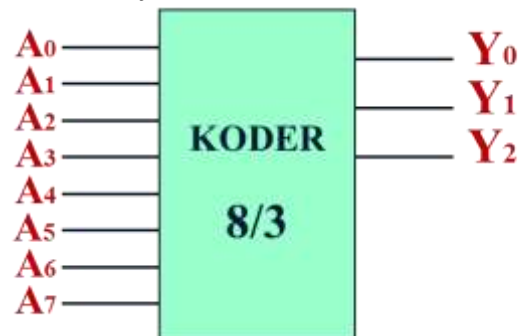
- Na ovom predmetu ćemo raditi sa koderom 8/3, što znači da ima 8 ulaza i 3 izlaza. Ako pogledamo bolje primetićemo da 8 ulaza predstavljaju cifre od 0 do 7 a tri izlaza predstavljaju 3 bita i kad bi sva 3 bita bili vrednosti 1 to znači da bi najveća moguća vrednost koja može da se predstavi na izlazu upravo broj 7. ($111_{(2)} = 7_{(10)}$)

- U datom trenutku, samo jedan od ulaza može biti aktivan tj. imati signal 1. U tom trenutku u zavisnosti od toga koji je ulaz aktivan, na izlazu se generiše binarna kombinacija koja odgovara rednom broju ulaza.

A ₇	A ₆	A ₅	A ₄	A ₃	A ₂	A ₁	A ₀	Y ₂	Y ₁	Y ₀
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

Kombinaciona tablica koder 8/3

Tabelu je najlakše zapamtiti tako što jedinice idu po dijagonali, a sa desne strane su klasični binarni 0-7 brojevi ILI A4 = 100, A5 = 101 itd.



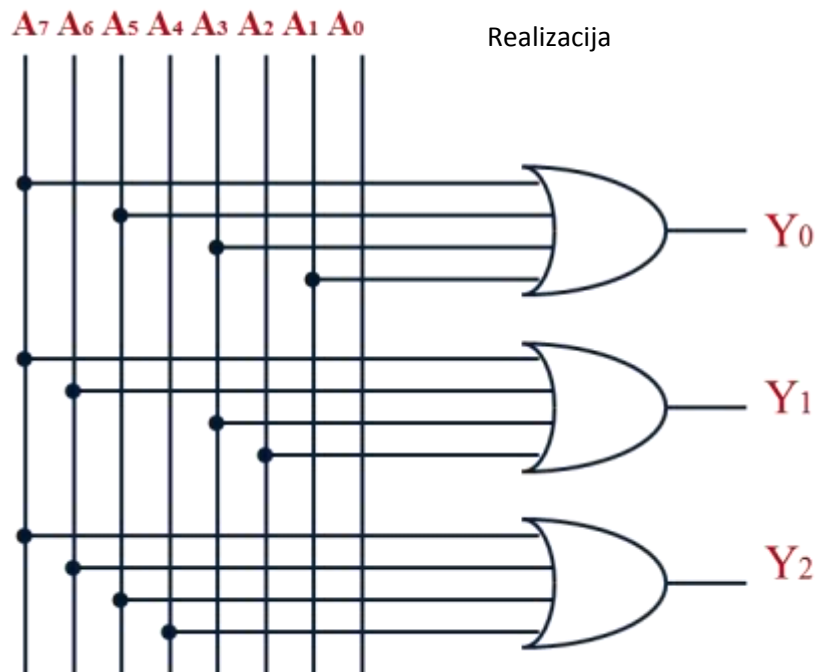
Funkcije izlaza koder 8/3

$$\begin{aligned} Y_0 &= A_1 + A_3 + A_5 + A_7 \\ Y_1 &= A_2 + A_3 + A_6 + A_7 \\ Y_2 &= A_4 + A_5 + A_6 + A_7 \end{aligned}$$

Y₀ je jedan kada su A₁, A₃, A₅ i A₇ jedan.

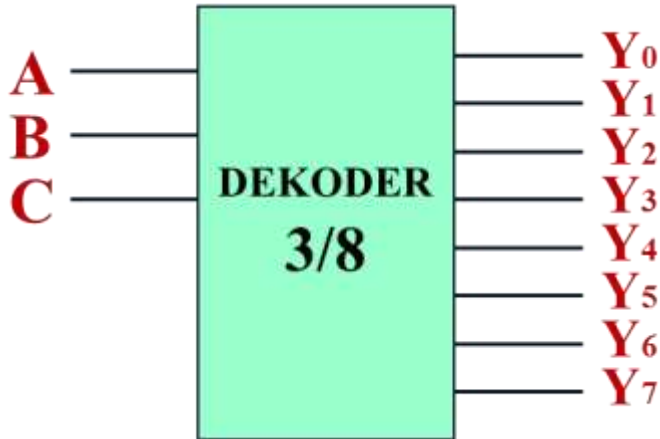
Y₁ je jedan kada su A₂, A₃, A₆ i A₇ jedan.

Y₂ je jedan kada su A₄, A₅, A₆ i A₇ jedan.



Dekoder

Obavlja funkciju dekodovanja binarno kodirane informacije dovedene na ulaz. Na izlazu se aktivira samo jedan izlaz koji odgovara ulaznoj kombinaciji.



$$Y_0 = A'B'C' = 000_{(2)} = 0_{(10)}$$

$$Y_1 = A'B'C = 001_{(2)} = 1_{(10)}$$

$$Y_2 = A'BC' = 010_{(2)} = 2_{(10)}$$

$$Y_3 = A'BC = 011_{(2)} = 3_{(10)}$$

$$Y_4 = AB'C' = 100_{(2)} = 4_{(10)}$$

$$Y_5 = AB'C = 101_{(2)} = 5_{(10)}$$

$$Y_6 = ABC' = 110_{(2)} = 6_{(10)}$$

$$Y_7 = ABC = 111_{(2)} = 7_{(10)}$$

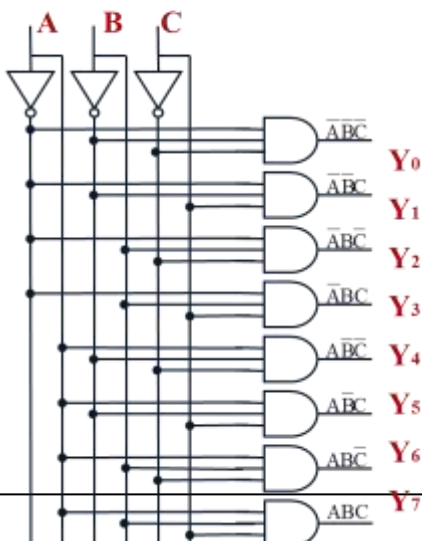
A	B	C		Y ₇	Y ₆	Y ₅	Y ₄	Y ₃	Y ₂	Y ₁	Y ₀
0	0	0		0	0	0	0	0	0	0	1
0	0	1		0	0	0	0	0	0	1	0
0	1	0		0	0	0	0	0	1	0	0
0	1	1		0	0	0	0	1	0	0	0
1	0	0		0	0	0	1	0	0	0	0
1	0	1		0	0	1	0	0	0	0	0
1	1	0		0	1	0	0	0	0	0	0
1	1	1		1	0	0	0	0	0	0	0

Kombinaciona tablica dekodera 3/8

Realizacija dekodera 3/8

$$\begin{aligned} Y_0 &= \overline{A}\overline{B}\overline{C} \\ Y_1 &= \overline{A}\overline{B}C \\ Y_2 &= \overline{A}B\overline{C} \\ Y_3 &= \overline{A}BC \\ Y_4 &= A\overline{B}\overline{C} \\ Y_5 &= A\overline{B}C \\ Y_6 &= AB\overline{C} \\ Y_7 &= ABC \end{aligned}$$

Funkcije izlaza dekodera 3/8



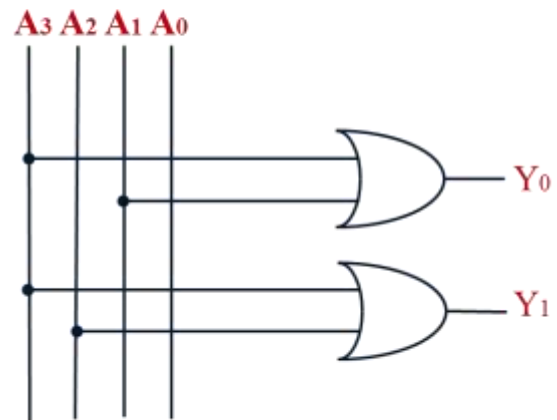
Primeri zadataka iz koda i dekodera

Realizovati koder 4/2.

A_3	A_2	A_1	A_0	Y_1	Y_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

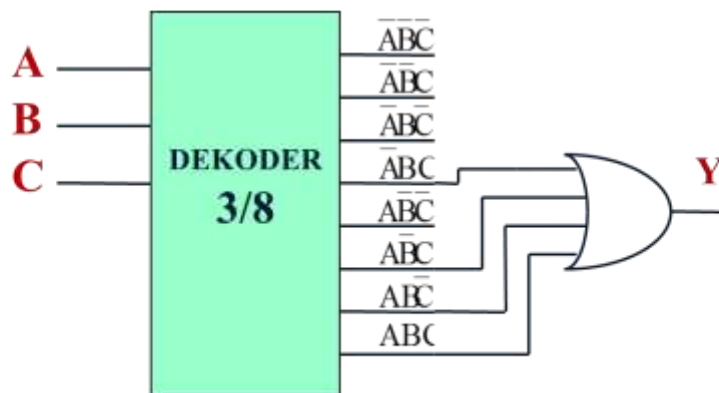
$$Y_0 = A_1 + A_3$$

$$Y_1 = A_2 + A_3$$



Zadatu logičku funkciju realizovati pomoću dekodera.

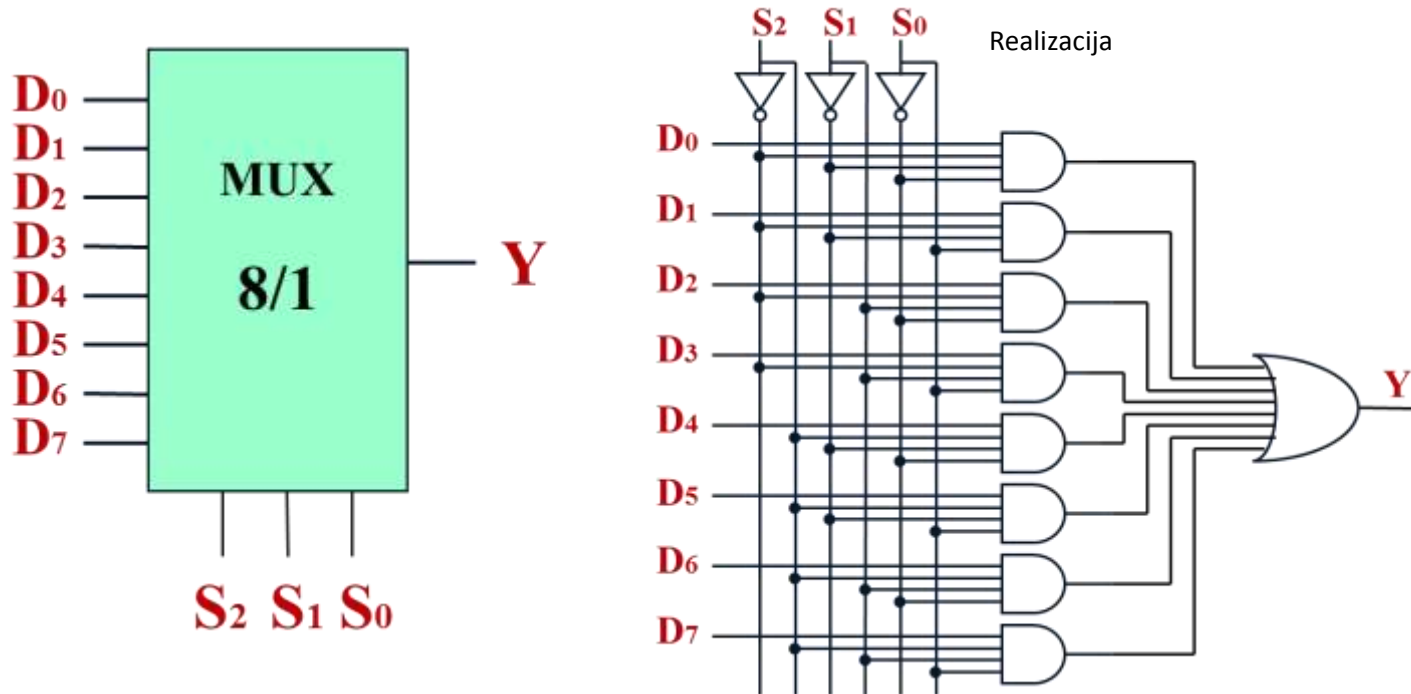
$$Y = \bar{A}BC + A\bar{B}C + ABC\bar{C} + ABC$$



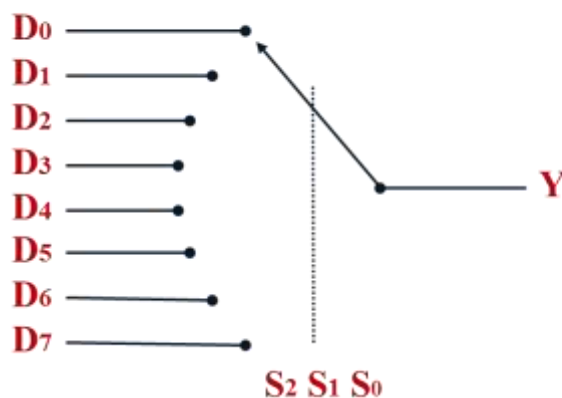
Multiplekser

Kombinaciona mreža (kao koder) sa više ulaza (n), jednim izlazom i više selekcionih signala (m). U zavisnosti od selekcionih signala SEL prekidač se postavlja u položaj koji odgovara jednom od ulaza, pa se izlaz direktno priključuje na taj ulaz. $n = 2^m$, te će u multiplexeru sa 8 ulaza biti 3 selekciona signala.

Multiplekser 8/1 ima 8 ulaza, 1 izlaz i 3 selekciona signala. Dovođenjem vrednosti na selekzione signale S_0 , S_1 i S_2 formira se birana kombinacija od 0-7 (000 – 111) koja predstavlja redni broj ulaza na koji se postavlja prekidač i samim tim taj ulaz se direktno prosleđuje na izlaz.



Kombinaciona tablica MUX 8/1



S_2	S_1	S_0	Y
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7

Funkcija izlaza MUX 8/1

$$Y = D_0 S_2 S_1 S_0 + D_1 S_2 S_1 \bar{S}_0 + D_2 S_2 \bar{S}_1 S_0 + \dots + D_7 S_2 S_1 S_0$$

Primer zadatka sa multiplekserom

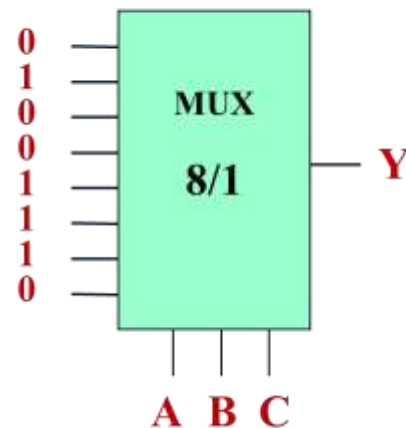
Zadatu logičku funkciju, predstavljenu sumom proizvoda, realizovati pomoću multipleksera.

$$Y = \overline{A}\overline{B}C + A\overline{B}\overline{C} + A\overline{B}C + ABC\overline{C}$$

Kombinaciona tablica

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Realizacija logičke funkcije



$$A'B'C = 001_{(2)} = 1_{(10)}$$

$$AB'C' = 100_{(2)} = 4_{(10)}$$

$$AB'C = 101_{(2)} = 5_{(10)}$$

$$ABC' = 110_{(2)} = 6_{(10)}$$

Permanentna memorija primer

Realizovati ROM za funkciju $x+2$ za dvobitne podatke

Tablica

A	B	Y_2	Y_1	Y_0
0	0	0	1	0
0	1	0	1	1
1	0	1	0	0
1	1	1	0	1

3

+2=

5

Jednačine

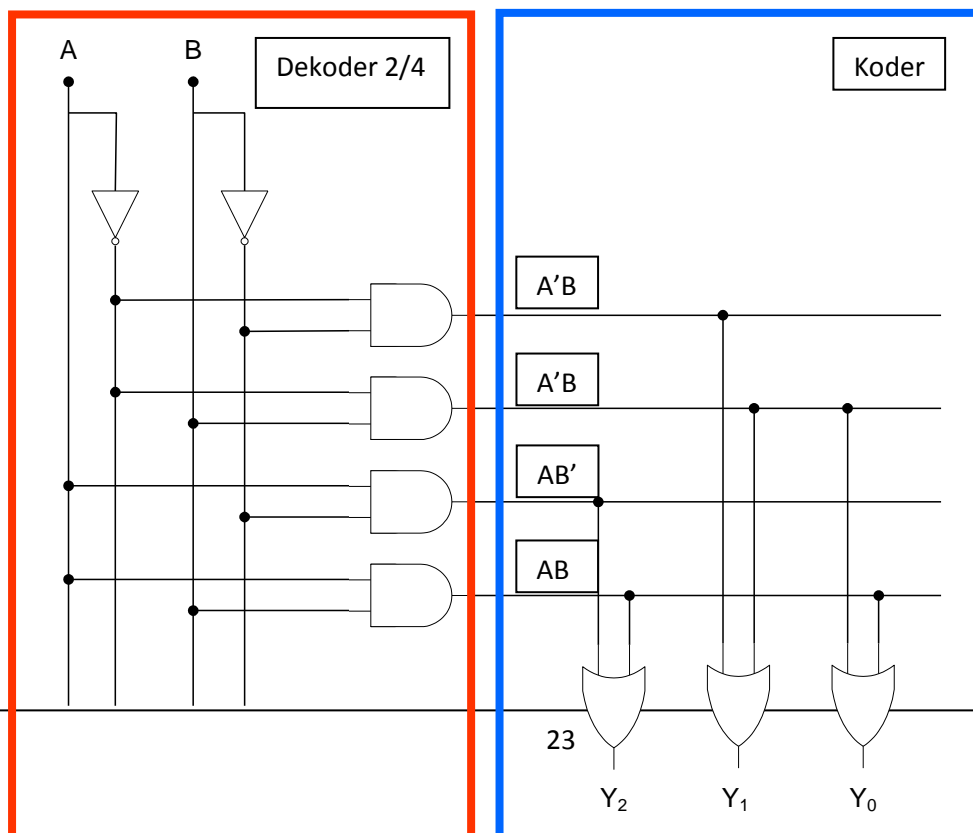
$$Y_2 = A \cdot \bar{B} + A \cdot B$$

$$Y_1 = \bar{A} \cdot \bar{B} + \bar{A} \cdot B$$

$$Y_0 = \bar{A} \cdot B + A \cdot B$$

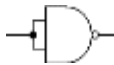
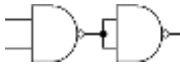
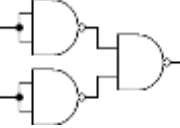
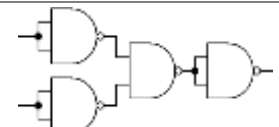
Dakle, kada X bude binarno 0, treba da se ispiše 2, kada je 1 treba da se ispiše 3 itd. Dvobitni podaci imaju ukupno 4 vrednosti (00, 01, 10, 11 tj. 0,1,2 i 3).

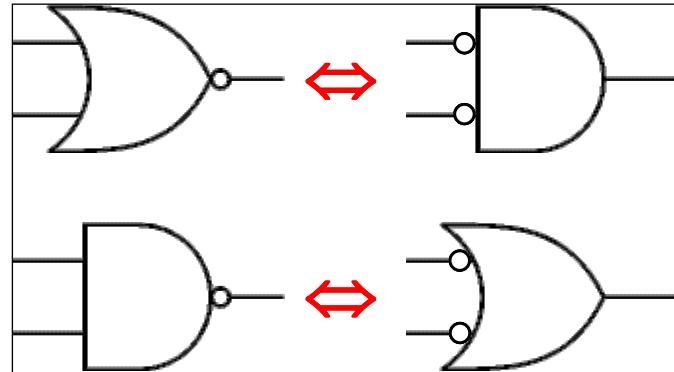
Jednačinu naučite napamet. Šema nam se sastoji iz dva dela. Kod dekodera pravimo dve grane A i B i pravimo 4 kola za sva moguća stanja ($00 = A'B'$, $01 = A'B$, $11 = AB$, $10 = AB'$), potom te rezultate kupimo u kodere tj. uzimamo rezultate koji su nam potrebni (kod Y_0 npr. nam trebaju sabirci $A'B$ i AB).



Inženjerska minimizacija

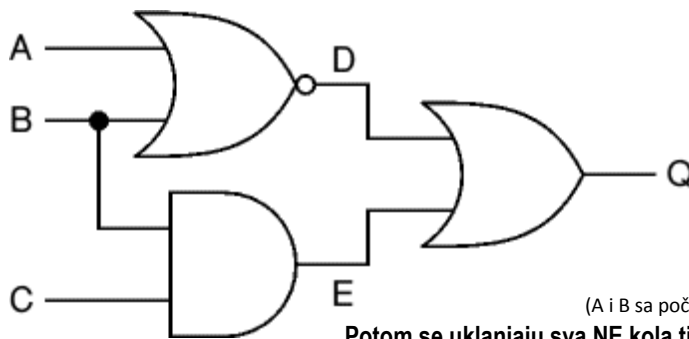
NI (NAND) ekvivalenti logičkih kola

Logičko kolo	Ekvivalent u NI (NAND) kolima
NE (NOT)	
I (AND)	
ILI (OR)	
NILI (NOR)	



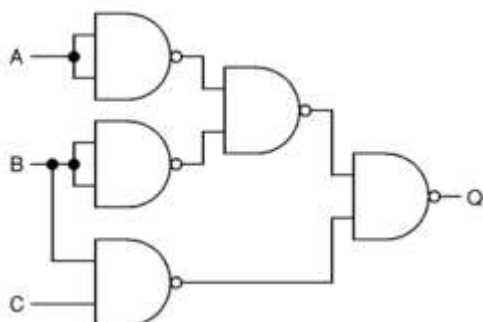
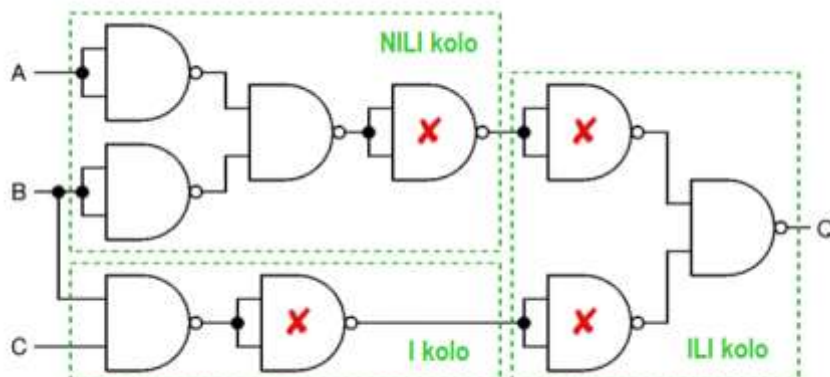
Kružić predstavlja inverter tj. suprotnu vrednost. Kružić na kraju I kola ga pretvara u NI kolo.

Primer - Originalni sistem ima 3 različita kola: NILI, I i ILI. To traži 3 logička kola (po jedno za svaki tip):



(A i B sa početka se ne nadovezuju na inverter pa se ne uklanjaju)

Potom se uklanjaju sva NE kola tj. invertori koja su uzastopna tj. 2 zaredom.



Minimizacija je izvršena.

IN zadatak analiza

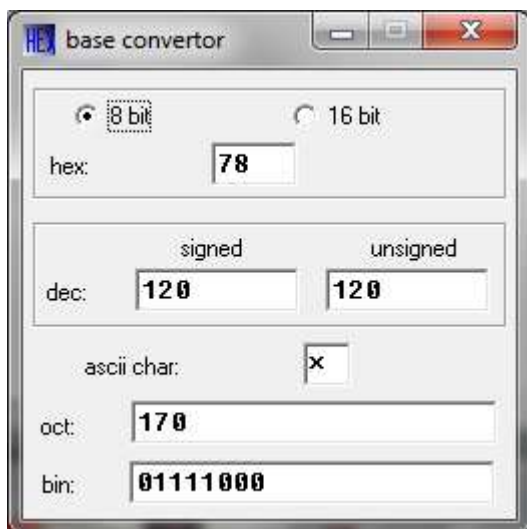
Kod mikroračunarskog sistema zasnovanog na procesoru 8088 port adresa 22h koristi se kao ulazni port za prihvatanje podataka o izmerenoj temperature.

Napisati sekvencu na asemblerskom jeziku koja kontinualno prihvta podatke sa port adrese 22h i kada temperature dostigne 120 °C upisati u registar BH ASCII kod znaka Y i pozvati program ALARM.

```

PONOVI:    IN      AL,22h ;pročitaj podatak o temperature sa porta #22h
           CMP     AL,120 ;da li je temperature 120 °C
           JNZ     PONOVI   ;ako nije, čitaj ponovo
           MOV     BH,'Y' ;temperature = 120 °C, napuni Y u BH
           CALL    ALARM    ;poyovi potprogram ALARM
           NOP

```



INTO zadatak - Analiziraj datu proceduru i dopuni/odgovori na pitanja.

INT 04 (Signed number overflow) – Interrupt on overflow

- INTO (interrupt on overflow).
- Ako se INTO postavi posle instrukcija aritmetike označenih brojeva IMUL ili ADD onda CPU aktivira INT ako je OF = 1.
- U slučaju OF=0 , INTO se ne izvršava

```
MOV AL, 64
MOV BL, 64
ADD AL, BL
INTO                ; OF = 1
```

```

+64
+64
-----
+128
```

- INTO izaziva CPU i "INT 04" i skače na fizičku lokaciju 00010H od IVT za CS : IP od ISR

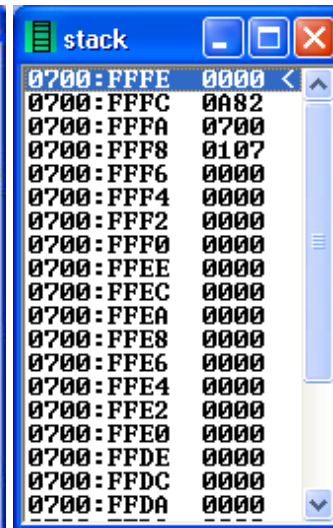
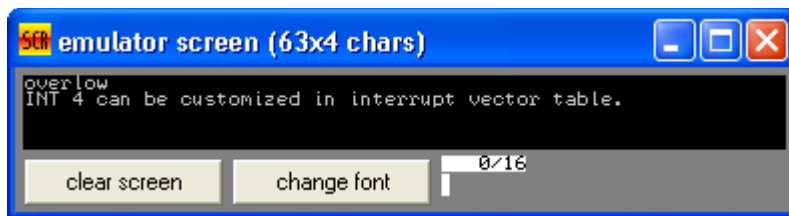
The screenshot shows three debugger windows:

- extended value viewer**: Shows the value of the AX register. The watch is set to AX. The value is 80 in hex, 10000000 in bin, 200 in oct, and 128 in decimal (signed). The signed value is -128.
- ALU - arithmetic & logic unit**: Shows the ALU unit with a table of flags. The flags are: CF=0, ZF=0, SF=1, OF=1, PF=0, AF=0, IF=1, DF=0.
- flags**: Shows the state of the flags. The flags are: CF=0, ZF=0, SF=1, OF=1, PF=0, AF=0, IF=1, DF=0.

INTO

The screenshot shows three debugger windows:

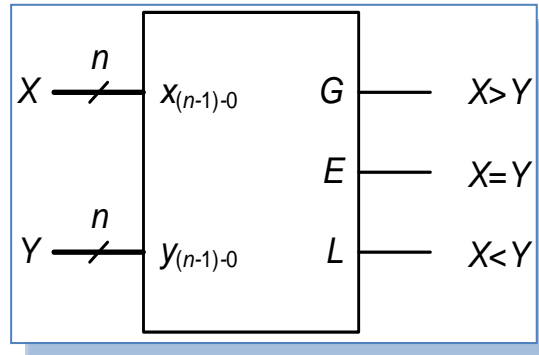
- flags**: Shows the state of the flags. The flags are: CF=0, ZF=0, SF=1, OF=1, PF=0, AF=0, IF=0, DF=0.
- lexical flag analyser**: Shows the state of the flags. The flags are: CF=0, ZF=0, SF=1, OF=1, PF=0, AF=0, IF=0, DF=0.
- stack**: Shows the stack. The stack is at address 0700:FFFE. The stack contains the following values: 0000, 0A82, 0700, 0107, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000, 0000.



Komparator

Komparator neoznačenih celih n-bitnih brojeva je mreža koja ima dva n-bitna ulaza (X i Y) i tri izlaza (G, E, L), od kojih samo jedan ima vrednost 1, dok su ostali na 0 u zavisnosti od odnosa ulaznih vrednosti:

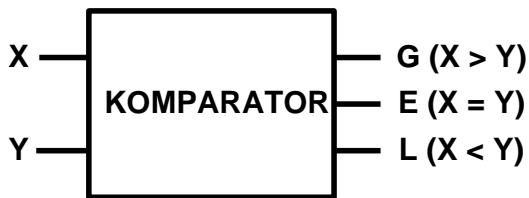
- **G = 1** ako je $X > Y$
- **E = 1** ako je $X = Y$
- **L = 1** ako je $X < Y$



$$G = X \cdot \bar{Y}$$

$$E = \bar{X} \cdot \bar{Y} + X \cdot Y = \overline{X \oplus Y} = \overline{X \cdot Y + X \cdot \bar{Y}}$$

$$L = \bar{X} \cdot Y$$



X	Y	$X > Y$	$X = Y$	$X < Y$
		G	E	L
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

