KOLEKCIJA ODABRANIH ISPITNIH PITANJA I ZADATAKA IZ PREDMETA "RAČUNARSKI SISTEMI"

- Potrebno je projektirati dvosmjerni brojač koji broji od 0 do 4 sa kontrolnim ulazom C koji za C = 0 broji unazad, a za C = 1 broji unaprijed, i koji po dostizanju krajnje vrijednosti zadržava tu vrijednost na izlazu sve do promjene kontrolnog ulaza C. Drugim riječima, za C = 0 brojač generira sekvencu binarnih brojeva $4, 3, 2, 1, 0, 0, 0$... (u binarnom kodu), a za C = 1 sekvencu binarnih brojeva $0, 1, 2, 3, 4, 4, 4$... Za projektiranje koristiti T flip-flopove i NAND logička kola.

- Potrebno je projektirati dvosmjerni brojač sa kontrolnim ulazom C koji za C = 0 generira sekvencu binarnih brojeva $0, 1, 2, 3, 4, 0, 1, 2, 3, 4$ ... (u binarnom kodu), a za C = 1 sekvencu binarnih brojeva $0, 4, 3, 2, 1, 0, 4, 3, 2, 1, 0$...
  a) Nacrtajte graf stanja ovog brojača.
  b) Projektirajte detaljnu strukturu ovog brojača ukoliko su na raspolaganju RS flip-flopovi, te osnovna logička kola.
  c) Ispitajte da li projektirani brojač posjeduje osobinu samostartiranja.

- Potrebno je projektirati dvosmjerni brojač sa kontrolnim ulazom C koji za C = 0 generira sekvencu binarnih brojeva $0, 1, 2, 0, 1, 2, ...$ (u binarnom kodu), a za C = 1 sekvencu binarnih brojeva $0, 2, 1, 0, 2, 1, 0$...
  a) Nacrtajte graf stanja ovog brojača.
  b) Projektirajte detaljnu strukturu ovog brojača ukoliko su na raspolaganju RS flip-flopovi, te NOR logička kola.

- Projektirajte dvosmjerni brojač sa kontrolnim ulazom C koji za C = 0 generira sekvencu binarnih brojeva $0, 1, 1, 2, 0, 1, 1, 2$... (u binarnom kodu) a za C = 1 sekvencu brojeva $0, 1, 2, 2, 0, 1, 2, 2$... Za projektiranje koristite T flip-flopove i osnovna logička kola.

- Projektirajte dvosmjerni brojač sa kontrolnim ulazom C koji za C = 0 generira sekvencu binarnih brojeva $0, 1, 0, 2, 0, 1, 0, 2$... (u binarnom kodu) a za C = 1 sekvencu brojeva $0, 2, 0, 1, 0, 2, 0, 1$... Za projektiranje koristite JK flip-flopove i osnovna logička kola.

- Projektirajte brojački sekvencijalni sklop tipa "djelitelj frekvencije sa 3", odnosno brojački sklop koji na izlazu generira sekvencu $0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, ...$ Za potrebe projektiranja koristite T flip-flopove, kao i osnovna logička kola.

- Potrebno je projektirati brojački sklop tipa "generator odnosa" bez ulaza (osim ulaza za taktne impulse), a koji na svom izlazu generira sekvencu $0, 0, 0, 0, 1, 0, 0, 0, 0, 1$...
  a) Nacrtajte graf stanja ovog sekvencijalnog sklopa.
  b) Projektirajte ovaj sklop koristeći JK flip-flopove, kao i osnovna logička kola.
  c) Da li je dobijeni sklop Mealyjevog ili Mooreovog tipa? Objasnite zašto.

- Potrebno je projektirati dvosmjerni brojač koji broji od 0 do 5 sa kontrolnim ulazom C koji za C = 0 broji unaprijed, a za C = 1 broji unazad, i koji po dostizanju krajnje vrijednosti zadržava tu vrijednost na izlazu sve do promjene kontrolnog ulaza C. Drugim riječima, za C = 0 brojač generira sekvencu binarnih brojeva $0, 1, 2, 3, 4, 5, 5, 5$... (u binarnom kodu), a za C = 1 sekvencu binarnih brojeva $5, 4, 3, 2, 1, 0, 0, 0$...
  a) Projektirajte ovaj brojač koristeći odgovarajući registar i odgovarajuću ROM memoriju, pri čemu je bitno navesti šta treba da bude upisano kao sadržaj ROM memorije.
  b) Nacrtajte detaljniju šemu projektovanog brojača, na kojoj se jasno vidi struktura upotrebljenog registra (preko D flip-flopova) i ROM memorije (preko dekodera i OR kola).

- Potrebno je projektirati sekvencijalni sklop brojačkog tipa sa kontrolnim ulazim C koji za C = 0 generiše sekvencu binarnih brojeva 0, 1, 0, 2, 0, 1, 0, 2, 0, 1, 0, 2... (u binarnom kodu), a za C = 1 sekvencu binarnih brojeva 0, 1, 1, 2, 0, 1, 1, 2, 0, 1, 1, 2...
  a) Nacrtajte graf stanja ovog sekvencijalnog sklopa.
  b) Projektirajte ovaj brojač koristeći odgovarajući registar i odgovarajuću ROM memoriju, pri čemu je bitno navesti šta treba da bude upisano kao sadržaj ROM memorije.
  c) Da li je ovaj sklop Mooreovog ili Mealyjevog tipa? Obrazložiti odgovor.

- Potrebno je projektirati dvosmjerni brojač sa kontrolnim ulazom C koji za C = 0 generira sekvencu binarnih brojeva $0, 1, 2, 3, 0, 1, 2, 3$... (u binarnom kodu), a za C = 1 sekvencu binarnih brojeva $0, 3, 2, 1, 0, 3, 2, 1$...
  a) Projektirajte ovaj brojač koristeći T flip-flopove i AND, OR i NOT logička kola.
  b) Projektirajte ovaj brojač koristeći odgovarajući registar i odgovarajuću ROM memoriju, pri čemu je bitno navesti šta treba da bude upisano kao sadržaj ROM memorije.
  c) Nacrtajte detaljniju šemu brojača projektovanog pod b), na kojoj se jasno vidi struktura upotrebljenog registra (preko D flip-flopova) i ROM memorije (preko dekodera i OR kola).

- Potrebno je projektirati dvosmjerni brojač koji broji od 0 do 3 sa kontrolnim ulazom C koji za C = 0 broji unaprijed, a za C = 1 broji unazad, i koji po dostizanju krajnje vrijednosti zadržava tu vrijednost na izlazu sve do promjene kontrolnog ulaza C. Drugim riječima, za C = 0 brojač generira sekvencu binarnih brojeva $0, 1, 2, 3, 3, 3, 3, 3$... (u binarnom kodu), a za C = 1 sekvencu binarnih brojeva $3, 2, 1, 0, 0, 0$... Na raspolaganju je jedan dvobitni registar i proizvoljan broj osnovnih logičkih kola.

- Koristeći 4-bitni registar i odgovarajuću ROM memoriju, projektirajte dvosmjerni brojač koji broji od 0 do 2 sa kontrolnim ulazom C koji za C = 0 broji unaprijed, a za C = 1 broji unazad. Drugim riječima, za C = 0 brojač generira sekvencu binarnih brojeva $0, 1, 2, 0, 1, 2$... (u binarnom kodu), a za C = 1 sekvencu binarnih brojeva $2, 1, 0, 2, 1, 0$... Obavezno naglasite koje brojeve treba upisati u ROM memoriju na koje adrese.

- Nacrtajte strukturu 4-bitnog paralelnog registra kod kojeg su ulazi i izlazi podataka spojeni zajedno, i objasniti kakva je prednost ovakvog registra u odnosu na registar kod kojeg su ulazi i izlazi podataka međusobno razdvojeni.

- Objasnite kakva je razlika između serijskih i paralelnih registara.

- Objasnite šta su to akumulirajući registri, zatim objasnite princip njihovog rada i navedite barem jedan primjer gdje se oni primjenjuju.

- Nacrtajte strukturu 5-bitnog pomjeračkog (šift) registra i objasnite princip njegovog rada na primjeru upisa i čitanja 5-bitne vrijednosti 11001 u ovaj registar.

- Objasnite šta su to serijsko-paralelni registri, kako su građeni i čemu služe.

- Nacrtajte strukturu 4-bitnog paralelnog registra kod kojeg su ulazi i izlazi podataka spojeni zajedno, i objasniti kakva je prednost ovakvog registra u odnosu na registar kod kojeg su ulazi i izlazi podataka međusobno razdvojeni.

- Objasnite šta su mikroinstrukcije, mikroprogrami i mikroprogramirani automati i u čemu je značaj mikroprogramiranih automata.

- Objasnite šta su PLD komponente.

- Nacrtajte strukturu RAM memorije koja može zapamtiti dva trobitna podatka i koja ima razdvojene linije za ulaz i izlaz podataka. Kao gradivne elemente za memoriju dozvoljeno je koristiti dekodere, multipleksere, D flip-flopove i osnovna logička kola.

- Nacrtajte strukturu RAM memorije koja može zapamtiti četiri dvobitna podatka i koja ima razdvojene linije za ulaz i izlaz podataka. Kao gradivne elemente za memoriju dozvoljeno je koristiti dekodere, multipleksere, registre i osnovna logička kola.

- Nacrtajte strukturu RAM memorije koja može zapamtiti dva dvobitna podatka i koja ima zajedničke linije za ulaz i izlaz podataka. Kao gradivne elemente za memoriju dozvoljeno je koristiti samo D flip-flopove i osnovna logička kola (i ništa drugo).

- Nacrtajte strukturu RAM memorije koja može zapamtiti dva dvobitna podatka i koja ima zajedničke linije za ulaz i izlaz podataka. Kao gradivne elemente za memoriju dozvoljeno je koristiti dekodere, multipleksere, D flip-flopove, elektronske prekidače i osnovna logička kola.

- Nacrtajte strukturu RAM memorije koja može zapamtiti dva trobitna podatka i koja ima zajedničke linije za ulaz i izlaz podataka. Kao gradivne elemente za memoriju dozvoljeno je koristiti dekodere, multipleksere, registre, elektronske prekidače i osnovna logička kola.

- Objasnite kakva je razlika između statičkih i dinamičkih RAM memorija.

- Objasnite ulogu registara MAR, MBR i PC u procesoru.

- Objasnite ulogu registara AC, IR i PC u procesoru, a zatim obrazložite zbog čega su potrebni pomoćni registri PA i PB ispred ulaza u aritmetičko-logičku jedinicu unutra procesora.

- Objasnite po čemu se MBR (MDR) registar procesora suštinski razlikuje od svih ostalih registara u procesoru.

- Registar MBR unutar procesora specifičan je u odnosu na ostale registre po svojoj dvosmjernosti, tj. osobini da mu ulaz i izlaz mogu međusobno zamijeniti mjesta, u zavisnosti od upravljačkog signala DIR. Koristeći D flip-flopove, osnovna logička kola i elektronske prekidače, sastavite šemu ovakvog registra.

- Objasnite u koliko faza se obavlja izvršavanje svake mašinske instrukcije i šta se dešava u svakoj od tih faza.

- Objasnite detaljno kako se odvija komunikacija između memorije i procesora ukoliko procesor upravo treba da sa adrese 50 pribavi mašinsku instrukciju čije se značenje može opisati kao “dobavi sadržaj adrese 70 iz memorije”. Pretpostavite da ova mašinska instrukcija zauzima jednu memorijsku lokaciju.

- Objasnite detaljno kako se odvija komunikacija između memorije i procesora ukoliko procesor upravo treba da sa adrese 45 pribavi mašinsku instrukciju čije se značenje može opisati kao “smjesti sadržaj prihvatnog registra (akumulatora) u memoriju na adresu 230”. Pretpostavite da ova mašinska instrukcija zauzima jednu memorijsku lokaciju.

- Objasnite strukturu jedne mašinske instrukcije procesora. Posebno istaknite šta je opkôd, a šta parametar instrukcije.

- Objasnite na koje se tri osnovne skupine dijele mašinske instrukcije i opišite ukratko osobine mašinskih instrukcija iz svake od tih skupina.

- Za razmotreni model hardverske strukture jednostavnog hipotetskog procesora koji prepoznaje četiri tipa mašinskih instrukcija, prateći graf stanja njegove upravljačke jedinice opišite čitav tok izvršavanja mašinske instrukcije čije bi se dejstvo moglo opisati frazom "dobavi sadržaj adrese *n* iz memorije i smjesti ga u prihvatni registar", počev od pribavljanja same instrukcije, preko njenog prepoznavanja i izvršavanja.

- Za razmotreni model hardverske strukture jednostavnog hipotetskog procesora koji prepoznaje četiri tipa mašinskih instrukcija, prateći graf stanja njegove upravljačke jedinice opišite čitav tok izvršavanja mašinske instrukcije čije bi se dejstvo moglo opisati frazom "smjesti sadržaj prihvatnog registra u memoriju na adresu *n*", počev od pribavljanja same instrukcije, preko njenog prepoznavanja i izvršavanja.

- Za razmotreni model hardverske strukture jednostavnog hipotetskog procesora koji prepoznaje četiri tipa mašinskih instrukcija, prateći graf stanja njegove upravljačke jedinice opišite čitav tok izvršavanja mašinske instrukcije čije bi se dejstvo moglo opisati frazom "saberi sadržaj prihvatnog registra sa sadržajem memorije na adresi *n* i smjesti ga ponovo u prihvatni registar", počev od pribavljanja same instrukcije, preko njenog prepoznavanja i izvršavanja.

- Za razmotreni model hardverske strukture jednostavnog hipotetskog procesora koji prepoznaje četiri tipa mašinskih instrukcija, prateći graf stanja njegove upravljačke jedinice opišite čitav tok izvršavanja mašinske instrukcije čije bi se dejstvo moglo opisati frazom "ukoliko je sadržaj prihvatnog registra negativan, nastavi izvršavanje od adrese *n*, u suprotnom nastavi izvršavanje od sljedeće instrukcije", počev od pribavljanja same instrukcije, preko njenog prepoznavanja i izvršavanja.

- Za razmotreni model hardverske strukture jednostavnog hipotetskog procesora koji prepoznaje četiri instrukcije sa mnemoničkim oznakama "`LOAD` *n*", "`STORE` *n*", "`ADD` *n*" i "`BNEG` *n*", objasnite u kratkim crtama šta se u procesoru tačno dešava prilikom izvršavanja instrukcije "`LOAD 100`" uz pretpostavku da se ova instrukcija nalazi u memoriji na adresi 30.

- Za razmotreni model hardverske strukture jednostavnog hipotetskog procesora koji prepoznaje četiri instrukcije sa mnemoničkim oznakama "`LOAD` *n*", "`STORE` *n*", "`ADD` *n*" i "`BNEG` *n*", objasnite detaljno sve korake koji se odigravaju između memorije i procesora prilikom izvršavanja sljedećeg programa:

```
0  LOAD   100
1  ADD    101
2  STORE  102
```

- Prevedite sljedeći mašinski program u ekvivalentni asemblerski program, uz pretpostavku da je program pisan za razmatrani hipotetski procesor koji razumije četiri instrukcije sa mnenoničkim oznakama "`LOAD`", "`STORE`", "`ADD`" i "`BNEG`" čiji su operacioni kodovi "00", "01", "10" i "11" respektivno:

```
0000000010001100
1000000010010110
0100001111101000
1100000000000000
```

- Objasnite šta znači kada se kaže da je neki procesor 32-bitni procesor.

- Objasnite šta znače pojmovi Little Endian i Big Endian.

- Prvi model jednostavnog hipotetskog procesora koji smo razmatrali razumije četiri instrukcije "`LOAD` *adresa*", "`STORE` *adresa*", "`ADD` *adresa*" i "`BNEG` *adresa*". Radi veće fleksibilnosti, instrukcija "`LOAD` *adresa*" je kasnije zamijenjena instrukcijom "`LC` *vrijednost*", koja u akumulator smješta navedenu vrijednost, a ne sadržaj navedene adrese.
  a) Nacrtajte kakve izmjene treba učiniti u grafu stanja upravljačke jedinice razmatranog procesora da bi on umjesto instrukcije "`LOAD`" prepoznavao i izvršavao opisanu instrukciju "`LC`". Radi jednostavnosti, pretpostavite da je parametar instrukcije "`LC`" uvijek nenegativan. Također pretpostavite da nova instrukcija "`LC`" ima isti mašinski kod kao što je imala instrukcija "`LOAD`" ("00*bbbbbbbbbbbbbb*", gdje je "*bbbbbbbbbbbbbb*" binarni kod broja koji je zadan kao parametar instrukcije "`LC`").
  b) Pokažite kako se nedostajuća instrukcija "`LOAD`" može simulirati pomoću instrukcija tako modificiranog procesora.

- Dat je sljedeći asemblerski program, za razmatrani hipotetski procesor:

```
PETLJA   LC      -5
         ADD     100
         BNEG    KRAJ
         STORE   100
         LC      –1
         BNEG    PETLJA
KRAJ     ...
```

  a) Ukoliko pretpostavimo da operacioni kodovi instrukcija "`LC`", "`STORE`", "`ADD`" i "`BNEG`" glase redom "00", "01", "10" i "11", prevedite ovaj asemblerski program u ekvivalentni mašinski program, uz pretpostavku da je program smješten u memoriju počev od lokacije 0.
  b) Ukoliko memorijska lokacija 100 na početku sadrži vrijednost 37, utvrdite kakva će biti vrijednost ove memorijske lokacije na kraju programa.
  c) Argumentirano utvrdite šta radi ovaj program, odnosno kakva je u općem slučaju zavisnost između početne i krajnje vrijednosti memorijske lokacije 100.

- Dat je sljedeći asemblerski program, za jednostavni procesor obrađivan na vježbama:

```
         LOAD    A
         STORE   REZULTAT
PETLJA   LOAD    B
         ADD     MINUS1
         BNEG    KRAJ
         STORE   B
         LOAD    REZULTAT
         ADD     MINUS1
         STORE   REZULTAT
         LOAD    MINUS1
         BNEG    PETLJA
KRAJ     ...
```

  a) Ako pretpostavimo da memorijske lokacije "`A`", "`B`" i "`MINUS1`" prije početka rada sadrže redom vrijednosti 5, 3 i –1, kolika će vrijednost biti u memorijskoj lokaciji "`REZULTAT`" po završetku rada programa?
  b) Prateći logiku rada ovog programa, utvrdite čemu ovaj program zapravo služi. Obrazložite odgovor.
  c) Ako ovaj procesor radi sa frekvencijom taktnih impulsa od 1 MHz (1000000 taktnih impulsa u sekundi), izračunajte koliko vremena traje izvršavanje ovog programa za podatke zadane pod a).

- Dat je sljedeći asemblerski program, za razmatrani hipotetski procesor:

```
         LOAD   PODATAK
         STORE  REZULTAT
PETLJA   LOAD   REZULTAT
         ADD    MINUS5
         BNEG   KRAJ
         STORE  REZULTAT
         LOAD   MINUS5
         BNEG   PETLJA
KRAJ     ...
```

  a) Ako pretpostavimo da memorijske lokacije "PODATAK" i "MINUS5" prije početka rada sadrže redom vrijednosti 10 i –5, kolika će vrijednost biti u memorijskoj lokaciji "REZULTAT" po završetku rada programa?

  b) Prateći logiku rada ovog programa, utvrdite čemu ovaj program zapravo služi. Obrazložite odgovor.

  c) Ako ovaj procesor radi sa frekvencijom kloka od 1 MHz (1000000 klok impulsa u sekundi), izračunati koliko vremena traje izvršavanje ovog programa za podatke zadane pod a). Uputa: poslužite se grafom stanja upravljačke jedinice i prebrojte neophodne cikluse.

- Dat je sljedeći asemblerski program, za razmatrani hipotetski procesor:

```
CIKLUS   LOAD   PODATAK
         ADD    MINUS3
         BNEG   KRAJ
         STORE  PODATAK
         LOAD   REZULTAT
         ADD    JEDAN
         STORE  REZULTAT
         LOAD   MINUS3
         BNEG   PETLJA
KRAJ     ...
```

  a) Ako pretpostavimo da memorijske lokacije "PODATAK", "REZULTAT", "MINUS3" i "JEDAN" prije početka rada sadrže redom vrijednosti 10, 0, –3 i 1, kolika će vrijednost biti u memorijskoj lokaciji "REZULTAT" po završetku rada programa?

  b) Prateći logiku rada ovog programa, utvrdite čemu ovaj program zapravo služi. Obrazložite odgovor.

- Dat je sljedeći asemblerski program, za razmatrani hipotetski procesor:

```
         LC      0
         STORE   REZULTAT
PETLJA   LOAD    N
         ADD     REZULTAT
         STORE   REZULTAT
         LC      -1
         ADD     N
         BNEG    KRAJ
         STORE   N
         LC      -1
         BNEG    PETLJA
KRAJ     ...
```

  a) Ako pretpostavimo da memorijska lokacija "N" na početku rada sadrži vrijednost 7, kolika će vrijednost biti u memorijskoj lokaciji "REZULTAT" po završetku rada programa?

  b) Prateći logiku rada ovog programa, utvrdite čemu ovaj program zapravo služi. Obrazložite odgovor.

- Dat je sljedeći asemblerski program, za razmatrani hipotetski procesor:

```
          LOAD    PODATAK
          STORE   BROJAC
          LC      -1
PETLJA    ADD     BROJAC
          BNEG    KRAJ
          STORE   BROJAC
          LOAD    REZULTAT
          ADD     PODATAK
          STORE   REZULTAT
          LC      -1
          BNEG    PETLJA
KRAJ      ...
```

  a) Ako pretpostavimo da memorijske lokacije "PODATAK" i "REZULTAT" prije početka rada sadrže redom vrijednosti 4 i 0, kolika će vrijednost biti u memorijskoj lokaciji "REZULTAT" po završetku rada programa?

  b) Prateći logiku rada ovog programa, utvrdite čemu ovaj program zapravo služi. Obrazložite odgovor.

- Dat je sljedeći asemblerski program, za razmatrani hipotetski procesor:

```
          LC      0
          STORE   Y
PETLJA    LOAD    X
          ADD     Y
          STORE   Y
          LC      -3
          ADD     X
          BNEG    KRAJ
          STORE   X
          LC      -1
          BNEG    PETLJA
KRAJ      ...
```

  a) Ako pretpostavimo da memorijske lokacija "X" prije početka rada sadrži vrijednost 13, kolika će vrijednost biti u memorijskoj lokaciji "Y" po završetku rada programa?

  b) Prateći logiku rada ovog programa, utvrdite čemu ovaj program zapravo služi. Obrazložite odgovor.

- Dat je sljedeći asemblerski program, za razmatrani hipotetski procesor:

```
          LOAD    A
          STORE   C
          LC      −1
PETLJA    ADD     B
          BNEG    KRAJ
          STORE   B
          LC      -1
          ADD     C
          STORE   C
          LC      −1
          BNEG    PETLJA
KRAJ      ...
```

  a) Ako pretpostavimo da memorijske lokacije "A" i "B" prije početka rada sadrže redom vrijednosti 6 i 3, kolika će vrijednost biti u memorijskoj lokaciji "C" po završetku rada programa?

  b) Prateći logiku rada ovog programa, utvrdite čemu ovaj program zapravo služi. Obrazložite odgovor.

- Dat je sljedeći asemblerski program, za razmatrani hipotetski procesor:

```
          LOAD   X
          STORE  Z
          LC     0
          STORE  Y
PETLJA    LC     -4
          ADD    Z
          BNEG   KRAJ
          STORE  Z
          LC     1
          ADD    Y
          STORE  Y
          LC     –1
          BNEG   PETLJA
KRAJ      ...
```

  a) Odredite kolika će vrijednost biti u memorijskim lokacijama "Y" i "Z" po završetku rada programa ukoliko pretpostavimo da memorijska lokacija "X" na početku rada sadrži vrijednost 25.

  b) Prateći logiku rada ovog programa, utvrdite čemu on zapravo služi, odnosno kakva je veza u općem slučaju između krajnjih vrijednosti lokacija "Y" i "Z" i početne vrijednosti lokacije "X". Obrazložite odgovor.

- Dat je sljedeći asemblerski program, za razmatrani hipotetski procesor:

```
          LC     0
          STORE  REZULTAT
          LC     -1
PETLJA    ADD    X
          BNEG   KRAJ
          STORE  X
          LC     -1
          ADD    Y
          BNEG   IZLAZ
          STORE  Y
          LC     -1
          BNEG   PETLJA
IZLAZ     LC     1
          STORE  REZULTAT
KRAJ      ...
```

  a) Ako pretpostavimo da memorijska lokacije "X" i "Y" na početku rada sadrže vrijednosti 5 i 4 respektivno, kolika će vrijednost biti u memorijskoj lokaciji "REZULTAT" po završetku rada programa?

  b) Ponovite istu analizu za slučaj kada su početne vrijednosti lokacija "X" i "Y" 4 i 5 respektivno.

  c) Prateći logiku rada ovog programa, utvrdite čemu ovaj program zapravo služi, odnosno kakva je zavisnost sadržaja memorijske lokacije "REZULTAT" na kraju rada programa od sadržaja memorijskih lokacija "X" i "Y" na početku rada programa.

- Dat je razmatrani hipotetski procesor koji razumije četiri instrukcije sa mnemoničkim oznakama "LC *vrijednost*", "STORE *adresa*", "ADD *adresa*", i "BNEG *adresa*".

  a) Sastavite asemblerski program za ovaj procesor koji dijeli sadržaj memorijske lokacije "A" sa 3, i smješta rezultat dijeljenja na lokaciju nazvanu "B" (dozvoljeno je i poželjno koristiti labele, kao i pomoćne lokacije). Po završetku programa, sadržaj lokacije "A" treba da ostane neizmijenjen.

  b) Šta se dešava sa ostatkom dijeljenja u programu koji ste upravo napisali, u slučaju da sadržaj memorijske lokacije "A" nije tačno djeljiv sa 3?

- Sastavite asemblerski program za razmatrani hipotetski procesor koji oduzima sadržaj lokacija "A" i "B", i ostavlja rezultat na lokaciji "C". Pretpostavite da je sadržaj lokacije "A" veći od sadržaja lokacije "B". Dozvoljeno je i poželjno koristiti labele, kao i pomoćne lokacije.

- Sastavite asemblerski program za razmatrani hipotetski procesor koji množi sadržaj dvije memorijske lokacije nazvane "A" i "B", i ostavlja rezultat na lokaciju nazvanu "C" (dozvoljeno je i poželjno koristiti labele). Pretpostavite da su oba sadržaja pozitivni brojevi. Po završetku programa, sadržaj lokacija "A" i "B" treba da ostane neizmijenjen.

- Sastavite asemblerski program za razmatrani hipotetski procesor koji kvadrira sadržaj lokacije "A" (tj. množi njen sadržaj sa samim sobom) i ostavlja rezultat u istoj lokaciji "A". Dozvoljeno je i poželjno koristiti labele, kao i pomoćne lokacije.

- Napišite asemblerski program za razmatrani hipotetski procesor koji u memorijsku lokaciju "Y" smješta kvadrat sadržaja lokacije "X" (npr. ako je sadržaj lokacije "X" na početku rada 4, nakon završetka rada sadržaj lokacije "Y" treba da bude 16). Po završetku rada sadržaj memorijske lokacije "X" treba da ostane neizmjenjen.

- Sastavite asemblerski program za razmatrani hipotetski procesor koji ispituje da li je sadržaj memorijske lokacije "PODATAK" neparan ili paran. Ukoliko je sadržaj lokacije "PODATAK" neparan, u lokaciju "NEPARAN" treba upisati jedinicu ("tačno"), a ukoliko je sadržaj lokacije "PODATAK" paran, u lokaciju "NEPARAN" treba upisati nulu ("netačno"). Dozvoljeno je i poželjno koristiti labele, kao i pomoćne lokacije.

- Sastavite asemblerski program za razmatrani hipotetski procesor koji dijeli sadržaj memorijske lokacije nazvane "X" sa 10 i ostavlja rezultat dijeljenja u memorijsku lokaciju nazvanu "Y", a ostatak dijeljenja u memorijsku lokaciju nazvanu "Z". Sadržaj memorijske lokacije "X" treba da ostane neizmijenjen. Dozvoljeno je i poželjno koristiti labele i pomoćne lokacije.

- Sastavite asemblerski program razmatrani hipotetski procesor koji množi sadržaj lokacije "A" sa 300, uz što je god moguće manji broj izvršenih operacija. Koristite tehniku zasnovanu na razlaganju broja 300 na zbir stepena broja 2.

- Sastavite asemblerski program za razmatrani hipotetski procesor koji računa vrijednost zbira S = 1 + 2 + 3 + ... + N gdje je N sadržaj istoimene memorijske lokacije. Rezultat treba da bude smješten u memorijsku lokaciju nazvanu "S". Dozvoljeno je i poželjno koristiti labele kao i pomoćne lokacije.

- Sastavite asemblerski program za razmatrani hipotetski procesor koji negira sadržaj memorijske lokacije "X", uz pretpostavku da je njen sadržaj na početku pozitivan. Na primjer, ukoliko je prije početka izvršavanja programa sadržaj lokacije "X" jednak 5, po završetku rada programa u lokaciji "X" treba da se nalazi broj –5. Dozvoljeno je i poželjno koristiti labele i pomoćne lokacije.

- Napišite asemblerski program za razmatrani hipotetski procesor koji u memorijsku lokaciju "Y" smješta negiranu vrijednost sadržaja lokacije "X" (npr. ako je sadržaj lokacije "X" na početku rada 4, nakon završetka rada sadržaj lokacije "Y" treba da bude –4). Po završetku rada sadržaj memorijske lokacije "X" treba da ostane neizmjenjen.

- Sastavite asemblerski program za razmatrani hipotetski procesor koji u memorijsku lokaciju "Z" smješta manju od dvije vrijednosti koje se nalaze u memorijskim lokacijama "X" i "Y". Sadržaj memorijskih lokacija "X" i "Y" na završetku programa treba da ostane neizmijenjen. Dozvoljeno je i poželjno koristiti labele.

- Objasnite na kojoj ideji se zasniva mogućnost rada procesora sa cijelim brojevima koji ne mogu stati u jednu memorijsku lokaciju.

- Objasnite na kojoj ideji se zasniva mogućnost rada procesora sa brojevima koji nisu cijeli.

- Objasnite šta su samomodificirajući programi i u kojim situacijama mogu biti korisni. Zatim objasnite na koji način je kod savremenih procesora izbjegnuta potreba za pisanjem samomodificirajućih programa.

- Prema teoremi Minskog, procesor koji poznaje svega dvije instrukcije ("`INC`" i "`DJNZ`") u stanju je principijelno obaviti bilo koji zadatak koji je u stanju uraditi ma koji procesor. Sastavite za ovakav procesor asemblerski program koji će udvostručiti sadržaj memorijske lokacije "100" (npr. ukoliko je sadržaj ove lokacije bio 7, nakon izvršavanja programa sadržaj iste lokacije treba da bude 14). Dozvoljeno je koristiti labele i pomoćne lokacije.

- Za minimalni procesor koji predviđa teorema Minskog, sastavite asemblerski program koji će pomnožiti sadržaj memorijske lokacije "`A`" sa 3 i smjestiti rezultat u memorijsku lokaciju "`B`". Sadržaj memorijske lokacije "`A`" treba da na kraju ostane neizmijenjen. Dozvoljeno je koristiti labele i pomoćne lokacije.

- Za minimalni procesor koji predviđa teorema Minskog, sastavite asemblerski program koji će sabrati sadržaj memorijskih lokacija "100" i "101" i smjestiti rezultat na lokaciju "102". Po završetku programa, lokacije "100" i "101" treba da ostanu prazne. Dozvoljeno je koristiti labele i pomoćne lokacije.

- Za minimalni procesor koji predviđa teorema Minskog, sastavite asemblerski program koji će oduzeti sadržaj memorijskih lokacija "100" i "101" (uz pretpostavku da je sadržaj lokacije "100" veći od sadržaja lokacije "101") i smjestiti rezultat na lokaciju "102". Po završetku programa, sadržaj lokacija "100" i "101" treba da ostane neizmijenjen. Dozvoljeno je i poželjno koristiti labele, kao i pomoćne lokacije.

- Za minimalni procesor koji predviđa teorema Minskog, sastavite asemblerski program koji će pomnožiti sadržaj memorijskih lokacija "100" i "101" i smjestiti rezultat na lokaciju "102". Dozvoljeno je koristiti labele i pomoćne lokacije.

- Za minimalni procesor koji predviđa teorema Minskog, sastavite asemblerski program koji će u memorijsku lokaciju "101" smjestiti polovinu sadržaja memorijske lokacije "100" uz pretpostavku da je sadržaj memorijske lokacije "100" paran. Dozvoljeno je i poželjno koristiti labele, kao i pomoćne lokacije.

- Potrebno je izračunati vrijednost aritmetičkog izraza

$$Y = A - \frac{A + \frac{A \cdot B \cdot C}{A + B + C}}{A - \frac{A \cdot B \cdot C}{A + B + C}}$$

gdje su A, B i C sadržaji izvjesnih memorijskih lokacija. Napišite kako bi izgledao asemblerski program koji računa vrijednost ovog izraza, ukoliko je na raspolaganju:

a) jednoadresni procesor, koji poznaje instrukcije "`ADD`", "`SUB`", "`MUL`", "`DIV`", "`LOAD`" i "`STORE`" (sve sa po jednim operandom);

b) nuladresni procesor, koji poznaje instrukcije "`ADD`", "`SUB`", "`MUL`" i "`DIV`" (bez operanada), kao i "`PUSH`" i "`POP`" (sa jednim operandom).

U oba slučaja, rezultat treba smjestiti na adresu Y. Eventualne pomoćne lokacije imenujte labelama P, Q, itd.

- Potrebno je izračunati vrijednost aritmetičkog izraza

$$Y = A - B \cdot \frac{A + \frac{B \cdot C}{B + C}}{A - \frac{B \cdot C}{B + C}}$$

gdje su A, B i C sadržaji izvjesnih memorijskih lokacija. Napišite kako bi izgledao asemblerski program koji računa vrijednost ovog izraza, ukoliko je na raspolaganju:

a) dvoadresni procesor, koji poznaje instrukcije "`ADD`", "`SUB`", "`MUL`", "`LOAD`" i "`STORE`" (sve sa po dva operanda);

b) nuladresni procesor, koji poznaje instrukcije "`ADD`", "`SUB`", "`MUL`" i "`DIV`" (bez operanada), kao i "`PUSH`" i "`POP`" (sa jednim operandom).

U oba slučaja, rezultat treba smjestiti na adresu Y. Eventualne pomoćne lokacije imenujte labelama P, Q, itd.

- Potrebno je izračunati vrijednost aritmetičkog izraza

$$Y = C - A \frac{D + \frac{A}{(B + C)(A - C)}}{D - \frac{A}{(B + C)(A - C)}}$$

gdje su A, B, C i D sadržaji izvjesnih memorijskih lokacija. Napišite kako bi izgledao asemblerski program koji računa vrijednost ovog izraza, ukoliko je na raspolaganju troadresni procesor koji poznaje instrukcije "`ADD`", "`SUB`", "`MUL`" i "`DIV`". Rezultat treba smjestiti na adresu Y. Eventualne pomoćne lokacije imenujte labelama P, Q, itd.

- Potrebno je izračunati vrijednost aritmetičkog izraza

$$Y = \frac{A + B}{C} - \frac{B}{A - C} \cdot \frac{B + \frac{C}{A + B}}{B - \frac{C}{A + B}}$$

gdje su A, B i C sadržaji izvjesnih memorijskih lokacija. Napišite kako bi izgledali asemblerski programi koji računaju vrijednost ovog izraza za dvoadresni procesor, koji poznaje instrukcije "`ADD`", "`SUB`", "`MUL`", "`DIV`" i "`MOVE`" (sa dva operandom), kao i za nuladresni procesor koji poznaje instrukcije "`ADD`", "`SUB`", "`MUL`", "`DIV`" (bez operanada) i instrukcije "`PUSH`" i "`POP`" (sa jednim operandom).

- Potrebno je izračunati vrijednost aritmetičkog izraza

$$Y = \frac{A}{B + C} - B \cdot \frac{A + \frac{C}{B}}{A - \frac{C}{B}}$$

gdje su A, B i C sadržaji izvjesnih memorijskih lokacija. Napišite kako bi izgledali asemblerski programi koji računaju vrijednost ovog izraza za jednoadresni procesor, koji poznaje instrukcije "`ADD`", "`SUB`", "`MUL`", "`DIV`", "`LOAD`" i "`STORE`" (sa jednim operandom), kao i za nuladresni procesor koji poznaje instrukcije "`ADD`", "`SUB`", "`MUL`", "`DIV`" (bez operanada) i instrukcije "`PUSH`" i "`POP`" (sa jednim operandom).

- Potrebno je izračunati vrijednost aritmetičkog izraza

$$Y = A - \frac{A + B + \frac{A}{B + C}}{A - B - \frac{A}{B + C}}$$

gdje su A, B i C sadržaji izvjesnih memorijskih lokacija. Napišite kako bi izgledao asemblerski program koji računa vrijednost ovog izraza, ukoliko je na raspolaganju:

a) jednoadresni procesor, koji poznaje instrukcije "`ADD`", "`SUB`", "`MUL`", "`DIV`", "`LOAD`" i "`STORE`" (sve sa po jednim operandom);

b) nuladresni procesor, koji poznaje instrukcije "`ADD`", "`SUB`", "`MUL`" i "`DIV`" (bez operanada), kao i "`PUSH`" i "`POP`" (sa jednim operandom).

U oba slučaja, rezultat treba smjestiti na adresu Y. Eventualne pomoćne lokacije imenujte labelama P, Q, itd.

- Potrebno je izračunati vrijednost aritmetičkog izraza

$$Y = A - B\frac{A + \frac{A \cdot B}{A + B - C}}{A - \frac{A \cdot B}{A + B - C}}$$

gdje su A, B i C sadržaji izvjesnih memorijskih lokacija. Napisati kako bi izgledao asemblerski program koji računa vrijednost ovog izraza, ukoliko je na raspolaganju:

a) dvoadresni procesor, koji poznaje instrukcije "`ADD`", "`SUB`", "`MUL`", "`LOAD`" i "`STORE`" (sve sa po dva operanda);

b) nuladresni procesor, koji poznaje instrukcije "`ADD`", "`SUB`", "`MUL`" i "`DIV`" (bez operanada), kao i "`PUSH`" i "`POP`" (sa jednim operandom).

U oba slučaja, rezultat treba smjestiti na adresu Y. Eventualne pomoćne lokacije imenujte labelama P, Q, itd.

- Potrebno je izračunati vrijednost aritmetičkog izraza

$$Y = A - B \cdot \frac{A + \frac{C}{B + C}}{A - \frac{C}{B + C}}$$

gdje su A, B i C sadržaji izvjesnih memorijskih lokacija. Napisati kako bi izgledao asemblerski program koji računa vrijednost ovog izraza, ukoliko je na raspolaganju:

a) troadresni procesor, koji poznaje instrukcije "`ADD`", "`SUB`", "`MUL`" i "`DIV`" (sa tri operanda);

b) nuladresni procesor, koji poznaje instrukcije `ADD`, `SUB`, `MUL` i `DIV` (bez operanada), kao i `PUSH` i `POP` (sa jednim operandom).

U oba slučaja, rezultat treba smjestiti na adresu Y. Eventualne pomoćne lokacije imenovati labelama P, Q, itd.

- Pretvorite sljedeće izraze zapisane u RPN zapisu u standardni (infiksni) zapis:

a) `A A B C + B C – * + A B C + B C – * – / –`

b) `A B C D E F + * + * +`

c) `A B C * D E * F + + +`

- Objasnite ukratko u čemu se sastoji suština indirektnog adresiranja i razloge zbog kojih je bilo neophodno uvesti ovakav način adresiranja.

- Sastavite asemblerski program koji udvostručava sadržaje svih memorijskih lokacija redom od adrese “100” do adrese “200”. Pretpostavite da procesor za koji pištete program, pored skupa instrukcija koje posjeduje razmatrani hipotetski procesor, posjeduje i instrukcije koje koriste indirektno adresiranje. Dozvoljeno je korištenje labela i pomoćnih lokacija.

- Sastavite asemblerski program koji sabira sadržaje svih memorijskih lokacija redom od adrese “300” do adrese “500” i ostavlja zbir u memorijskoj lokaciji “1000”. Pretpostavite da procesor za koji pišete program, pored skupa instrukcija koje posjeduje razmatrani hipotetski procesor, posjeduje i instrukcije koje koriste indirektno adresiranje. Dozvoljeno je korištenje labela i pomoćnih lokacija.

- Ukoliko su sadržaji registra R1 te memorijskih lokacija 100, 200, 300 i 400 redom 300, 200, 400, 500 i 100, odgovorite šta će biti u akumulatoru nakon izvršavanja svake od sljedećih 5 asemblerskih instrukcija:

  a) `LOAD #100` b) `LOAD 100` c) `LOAD R1` d) `LOAD (R1)` e) `LOAD @100`

  Obavezno obrazložite odgovor.

- Objasnite ulogu ulazno-izlaznih veznih sklopova i ukratko objasnite njihovu strukturu. Posebno istaknite šta je međusklop i zbog čega on najčešće nije čisto digitalni uređaj.

- Objasnite koja je razlika između memorijski mapiranog i neovisno mapiranog ulaza/izlaza. Istaknite prednosti i nedostatke jednog i drugog načina. Posebno istaknite u čemu je razlika u povezivanju procesora, memorije i ulazno-izlaznih uređaja pri korištenju ova dva načina.

- Objasnite osnovne činjenice o kontrolerima tastature i načinu povezivanja tastature sa kontrolerima tastature.

- Objasnite šta je matrični spoj tastera i koje su mu prednosti i nedostaci u odnosu na klasično spajanje.

- Objasnite osnovne činjenice o kontrolerima ekrana. Posebno objasnite smisao i ulogu video memorije, te razliku između tekstualnih i grafičkih kontrolera ekrana.

- Objasnite u osnovnim crtama komunikaciju između procesora i eksternih memorija. Posebno istaknite kako se upravlja kontrolerima diskova i čemu služi i kako se realizira DMA prenos.

- Opišite u kratkim crtama građu diskova.

- Opišite u kratkim crtama osnovne činjenice o kontrolerima diskova.

- Objasnite kakva je razlika između prozivke (polinga) i prekida (interapta) i na koji način se prekidi realiziraju.

- Neki računar koristi video kontroler koji prikazuje sliku u tekstualnom režimu formata 80×25 znakova. Video memorija počinje na adresi “10000”, a svakom znaku odgovara po jedna adresa (registar) u video memoriji, koja čuva ASCII šifru odgovarajućeg znaka na ekranu. Sastavite asemblerski program koji će čitav ekran popuniti zvjezdicama, znajući da ASCII šifra znaka za zvjezdicu “*” iznosi 42. Pretpostavite da procesor za koji se piše program poznaje instrukcije kao i razmatrani hipotetski procesor, uz dodatak instrukcija “`LDIND`” i “`STIND`” koje koriste indirektno adresiranje.

- Neki računar koristi video kontroler koji prikazuje sliku u tekstualnom režimu formata 80×25 znakova. Video memorija počinje na adresi "5000", a svakom znaku odgovaraju po dvije adrese (registra) u video memoriji, od kojih prvi čuva ASCII šifru odgovarajućeg znaka na ekranu, a drugi informacije o boji (atribut) znaka. Napišite kako bi principijelno mogao izgledati asemblerski program koji će čitav ekran popuniti znacima "#" bijelom bojom, znajući da ASCII šifra tog znaka iznosi 35, a atribut za bijelu boju je 7. Pretpostavite da procesor za koji se piše program poznaje instrukcije kao i razmatrani hipotetski procesor, uz dodatak instrukcija "`LDIND`" i "`STIND`" koje koriste indirektno adresiranje.

- Napišite kako bi principijelno izgledao asemblerski program koji prebacuje blok sa diska dug 1024 bajta koji se nalazi u traci 12 i sektoru 9 u radnu memoriju počevi od adrese 1000, uz pretpostavku da kontroler diska ne koristi DMA prenos. Pretpostavite da procesor za koji se piše program poznaje instrukcije kao i razmatrani hipotetski procesor, uz dodatak instrukcija "`LDIND`" i "`STIND`" koje podržavaju indirektno adresiranje. Također, pretpostavite da komanda kontrolera za upis na disk ima šifru "1". Adrese registara kontrolera diska imenujte odgovarajućim labelama.

- Opišite kako bi principijelno mogao izgledati asemblerski program koji prebacuje blok sa diska koji se nalazi u traci 7 i sektoru 5 u memoriju na adresu 2000, uz pretpostavku da kontroler diska koristi DMA prenos.

- Objasnite šta je firmver i zbog čega svaki računarski sistem mora imati i RAM i ROM memoriju.

- Objasnite kakva je računara između programabilnih računara i računara specijalne namjene i čime je određeno da li će neki računar biti programabilni računar ili računar specijalne namjene.

- Objasnite šta je operativni sistem i koja je njegova uloga. Također objasnite ulogu monitora kao primitivnog operativnog sistema.

- Objasnite ukratko od kojih se cjelina sastoji operativni sistem, i koja je uloga svake od navedenih cjelina.

- Objasnite šta su to drajveri.

- Objasnite šta je preključivanje i kako se ono realizira. Kojem dijelu operativnog sistema pripada sloj za upravljanje preključivanjima?

- Objasnite koja je uloga sloja za upravljanje datotekama u operativnim sistemima i osnovne ideje koje leže iza njegove realizacije. Posebno istaknite šta je FAT tabela i koja joj je uloga. Kojoj cjelini operativnog sistema pripada ovaj sloj?

- Objasnite pojam procesa i ulogu sloja za upravljanje procesima u operativnim sistemima. Posebno istaknite šta se podrazumjeva pod višezadačnim (multitasking) operativnim sistemima. Kojem dijelu operativnog sistema pripada ovaj sloj?

- Objasnite ulogu sloja za upravljanje memorijom u operativnim sistemima. Posebno objasnite mehanizam virtualne memorije, i zbog čega količina raspoložive RAM memorije može uticati na brzinu rada računara. Kojem dijelu operativnog sistema pripada ovaj sloj?

- Opišite ukratko kako se vrši učitavanje operativnog sistema prilikom uključivanja računara u slučajevima kada se operativni sistem ne nalazi cjelokupno u ROM memoriji. Koje su prednosti realizacija kod kojih se operativni sistem ne nalazi cjelokupno u ROM memoriji u odnosu na situaciju kada je čitav operativni sistem u ROM-u?

- Objasnite šta je to BIOS i šta on tipično sadrži kod današnjih modernih računara.

- Objasnite ukratko kako teče postupak prevođenja programa iz nekog višeg ne-mašinskog programskog jezika u mašinski jezik.