Uvod u organizaciju i arhitekturu računara 2

1 Asemblersko programiranje u Intel 64 arhitekturi - ZADACI

Pročitati izvod iz dokumentacije *intel64.pdf*. Za svaku od sledećih funkcija potrebno je napisati i program u C-u koji poziva datu funkciju.

1.1 Čas1 - uvod

1. **hello** - Napisati program u asembleru koji ispisuje pozdravnu poruku na izlaz.

2. **saberi** - Napisati funkciju *int suma*(*int a*, *int b*) koja sabira dva cela broja a i b.

1.2 Čas2 - uslovni operatori, binarni operatori

3. **oduzmi** - Napisati funkciju *int razlika*(*int a*, *int b*) koja oduzima dva cela broja a i b.

4. **negacija** - Napisati funkciju *unsigned negacija*(*unsigned x*) koja računa bitovku negaciju broja x.

5. **domaci** - Napisati funkciju *unsigned izraz*(*unsigned a*, *unsigned b*, *unsigned c*) koja izračunava izraz $(4a - b + 1)/2 + c/4$, za prosleđene a, b, i c kao argumente.

6. **domaci** - Napisati funkciju *void aritmetika*(*int a*, *int b*) koja za prosleđena dva operanda računa sledeće: zbir, razliku, proizvod, količnik, suprotnu vrednost, bitovsku konjunkciju, bitovsku disjunkciju, bitovsku negaciju, prvi broj šiftovan u levo za vrednost drugog operanda, prvi broj šiftovan aritmetički u desno za vrednost drugog operanda. Rezultate ispisivati pozivanjem funkcije *printf*.

7. **max** - Napisati funkciju *int max*(*int a*, *int b*) koja vraća maksimum argumenata a i b.

8. **domaci** - Napisati funkciju *int min*(*int a*, *int b*) koja računa minimum dva broja a i b.

9. **deljiv_4** - Napisati funkciju *int deljiv_4*(*int x*) koja vraća 1 ako je argument x deljiv sa 4, a 0 ako nije.

10. **prestupna** - Napisati funkciju *unsigned prestupna*(*unsigned x*) koja proverava da li je godina data argumentom x prestupna (ako je deljiva sa 4; izuzetak tome su godine deljive sa 100; izuzetak tom izuzetku su godine deljive sa 400). Funkcija treba da vraća 1 ako godina jeste prestupna, u suprotnom treba da vraća 0.

1.3 Čas3 - petlje

11. **suma** - Napisati funkciju *unsigned suma*(*unsigned n*) koja računa sumu prvih n prirodnih brojeva, počev od 1.

12. **suma_parnih** - Napisati funkciju *unsigned suma_parnih*(*unsigned n*) koja računa sumu prvih n prirodnih brojeva koji su parni, počev od 1.

13. **suma_cifara** - Napisati funkciju *unsigned suma_cifara*(*unsigned n*) koja računa sumu cifara dekadnog broja n.

14. **broj_jedinica** - Napisati funkciju *unsigned broj_jedinica(unsigned n)* koja vraća broj jedinica u binarnom zapisu broja n.

15. **domaci** - Napisati funkciju *int deljiv_16(int n)* koja vraća 1 ako je argument n deljiv sa 16, a 0 ako nije.

16. **domaci** - Napisati funkciju *unsigned min(unsigned a, unsigned b, unsigned c)* koja računa minimum argumenata.

17. **domaci** - Napisati funkciju *unsigned najveca_cifra(unsigned x)* koja računa najveću cifru dekadnog broja x.

18. **domaci** - Napisati funkciju *unsigned faktorijel(unsigned x)* koja vraća faktorijel argumenta x.

19. **nzd** - Napisati funkciju *unsigned nzd(unsigned a, unsigned b)* koja računa NZD svojih argumenata Euklidovim algoritmom ($nzd(a,0) = a; nzd(a,b) = nzd(b, a\%b)$).

20. **ojler** - Napisati funkciju *unsigned ojler(unsigned n)* koja implementira Ojlerovu funkciju ($f(n)$ jednaka je broju brojeva k koji su manji od n i uzajamno prosti sa n). Koristiti prethodnu funkciju za računanje nzd.

21. **prost** - Napisati funkciju *unisigned prost(unsigned n)* koja vraća 1 ukoliko je broj n prost, a inače vraća 0.

1.4 Čas4 - pokazivači i nizovi

22. **zameni** - Napisati funkciju *void zameni(int* a, int* b)* koja zamenjuje vrednosti na prosleđenim lokacijama a i b.

23. **kolicnik** - Napisati funkciju *void kolicnik(unsigned a, unsigned b, unsigned* k, unsigned* o)* koja računa količnik i ostatak dva broja a i b i upisuje te vrednosti preko pokazivača. Brojevi su dati kao prvi i drugi argument, dok su pokazivači dati kao treći i četvrti argument.

24. **bitovska_aritmetika** - Napisati funkciju *void bitovska_aritmetika(unsigned a, unsigned b, unsigned* k, unsigned* d, unsigned* e, unsigned* n)* koja za prva dva prosledjena argumenta a i b izračunava bitovsku konjunkciju, bitovsku disjunkciju, bitovsku ekskluzivnu disjunkciju i negaciju prvog argumenta. Ove četiri vrednosti treba proslediti preko pokazivača k, d, e i n.

25. **suma_niza** - Napisati funkciju *int suma_niza(int* A, int n)* koja računa sumu elemenata niza. Adresa niza je data kao prvi argument, dok je broj članova niza dat kao drugi argument.

26. **najveci** - Napisati funkciju *int najveci(int* A, int n)* koja računa najveći element u nizu. Adresa niza je data kao prvi argument, dok je broj članova niza dat kao drugi argument.

27. **obrni** - Napisati funkciju *void obrni(int* A, int n)* koja vrši obrtanje niza. Adresa niza je data kao prvi argument, dok je broj članova niza dat kao drugi argument.

1.5 Čas5 - nizovi vežbanje

28. **fibonaci** - Napisati funkciju *void fibonaci(int n, int* A)* koja prvih n Fibonačijevih brojeva smešta u niz F. Adresa niza je data kao drugi argument, dok je broj n dat kao prvi argument.

29. **izdvoji_proste** - Napisati funkciju *int izdvoji_proste(int* A, int n, int* B)* koja iz niza brojeva A dužine n izdvaja samo proste brojeve i smešta ih u drugi niz B. Veličinu drugog niza vratiti kao povratnu vrednost funkcije.

30. **domaci** - Napisati funkciju *int zbir_apsolutnih(int* A, int n)* koja računa zbir apsolutnih vrednosti brojeva u nizu. Adresa niza je data kao prvi argument, dok je broj članova niza dat kao drugi argument.

31. **domaci** - Napisati funkciju *int suma_razlika(int* a, int n)* koja računa sumu razlika $|a[1]-a[0]| + |a[2]-a[1]| + ... + |a[n-1]-a[n-2]|$ za dati niz a. Adresa niza je data kao prvi argument, dok je broj članova niza dat kao drugi argument.

32. **domaci** - Napisati funkciju *int najveci_3(int* A, *int* n) koja pronalazi najveći element u nizu A dužine n koji je deljiv sa 3. Adresa niza je data kao prvi argument, dok je broj članova niza dat kao drugi argument.

33. **domaci** - Napisati funkciju *int deljiv(int* a, *int* b) koji za prosleđena dva pozitivna a i b broja vraća vrednost 1 ukoliko je prvi broj deljiv drugim brojem, u suprotnom vraća vrednost 0. Napisati zatim funkciju *void izdvoji_deljive(int** A, *int* na, *int** B, *int** nb) koja iz datog niza A izdvaja u drugi niz B brojeve koji su deljivi sa sedam koristeći prethodnu funkciju. Adresa niza je data kao prvi argument, dok je broj članova niza dat kao drugi argument. Adresa drugog niza je data kao treći argument, dok veličinu drugog niza treba upisati na adresu prosleđenu kao četvrti argument.

34. **savrsen_stepen** - Napisati funkciju *int savrsen_stepen(unsigned* n, *unsigned** m, *unsigned** k) koja vraća 1 ako postoje $m, k >= 2$ takvi da $m^k = n$ a 0 inače. Ukoliko postoje traženi m i k, vratiti ih preko pokazivača m i k.

35. **izbaci_neparne** - Napisati funkciju *void izbaci_neparne(long** niz, *unsigned** $duzina$) koja izbacuje elemente koji imaju neparnu vrednost iz niza koji je dat prvim argumentom i čija je dužina u drugom argumentu.

1.6 Čas6 - vežbanje

36. **savrsen** - Napisati funkciju *int savrsen(int* x) koja ispituje da li je broj x savršen. Broj je savršen ako je jednak zbiru svojih pravih delilaca, npr. 6 je takav.

37. **faktorizacija** - Napisati funkciju *int* $faktorizacija$(*int* x, *int** A, *int** B) koja radi faktorizaciju broja: za dati broj x u niz A upisati njegove proste faktore, a u niz B odgovarajuće višestrukosti. Veličinu nozova A i B vratiti kao povratnu vrednost. Npr. $120 = 2^3 * 3 * 5$, pa u niz A treba upisati 2, 3, 5, a u niz B redom 3, 1, 1 (višestrukosti faktora 2, 3, 5 respektivno).

38. **clan_niza** - Napisati program *int clan_niza(int)* koji izračunava n-ti član niza koji je dat rekurentnom formulom $A_0 = 1$, $A_n = 4 * A_{n-1} + 3$, $n >= 1$.

39. **smesti** - Napisati funkciju *int vrednost(int* x) koji za ceo broj x izračunava vrednost $f(x) = min\{24, x^2\}$. Napisati zatim funkciju *void smesti(int** X, *int* nx, *int** Y, *int** ny) koji od niza X dužine nx formira novi niz Y koji se sastoji od vrednosti funkcije $f(x)$ primenjene na sve članove niza X čija je vrednost parna. Veličinu niza Y treba vratiti kao povratnu vrednost. Neka je niz $X = 2, 3, 6, 1, -7, 8, -4, 0$. Posle pokretanja programa niz Y treba da ima članove $Y = 4, 24, 24, 16, 0$.

1.7 Čas7 - vežbanje

40. **napravi_niz** - Napisati funkciju *int sadrzi(int** A, *int* n, *int* c) koji za dati niz A različitih elemenata i broj c, proverava da li niz A sadrži član koji ima vrednost c. Ukoliko ne sadrži, funkcija vraća negativnu vrednosti, dok u suprotnom vraća poziciju tog člana. Adresa niza A, dužina niza i broj c su dati kao argumenti funkcija. Napisati zatim funkciju *void napravi_niz(int** A, *int** B, *int* n, *int** C, *int** nc) koji na osnovu dva niza A i B, dužine n, menja niz C, dužine nc, na sledeći način: za svaki element $C[i]$ niza C, ukoliko postoji element $A[j]$ niza A takav da je $A[j] = C[i]$, element $C[i]$ se zamenjuje sa $B[j]$. U suprotnom, $C[i]$ ostaje nepromenjen. Pretpostaviti da su svi elementi niza A različiti, da su nizovi A i B jednakih dužina. Koristiti funkciju iz prvog dela zadatka za proveru da li se element $C[i]$ nalazi u nizu A. Primer, ako su nizovi $A: 42, 35, 90, 21$; $B: 84, 7, 30, 22$; i $C: 32, 35, 42, 9, 12, 0$, po završetku programa niz C treba da izgleda ovako $32, 7, 84, 9, 12, 0$.

41. **n_ti_clan** - Napisati funkciju *int n_ti_clan(int* a, *int* b, *int* n) koji računa n-ti clan niza S_n po sledećoj rekurentnoj formuli: $S_1 = a$, $S_2 = b$ i $S_n = 32 * S_{n-1} - S_{n-2}$.

42. **s_niz** - Napisati funkciju *int s_niz(int** S, *int* n, *int* a, *int* b) koji računa članove niza S po sledećoj rekurentnoj formuli: $S_1 = a$, $S_2 = b$ i $S_n = 32 * S_{n-1} - S_{n-2}$.

43. **vrednost** - Napisati funkciju *unsigned long vrednost(unsigned long* n) koji za nenegativni celobrojni parametar n izračunava vrednost $2^0 * 0 * 1 + 2^1 * 1 * 2 + 2^2 * 2 * 3 + ... + 2^{n-1} * (n-1) * n + 2^n * n * (n+1)$.

1.8 Čas8 - vežbanje

44. **napravi** - Napisati funkciju *int izracunaj*(*int x*, *inty*, *intk*) koja za tri pozitivna celobrojna argumenta x, y i k izračunava proizvod $2^k * (x–1) * (y–1)$. Napisati zatim funkciju *void napravi*(*int* X*, *int* Y*, *int n*, *int* Z*) koja od dva niza X i Y prirodnih brojeva dužine n formira niz Z, tako da je $Z_i = 2^k * (X_i–1)(Y_i–1)$, pri čemu je $k = 1$ ako je $X_i + Y_i$ neparan broj, a inače je $k = 0$.

45. **izracunaj2** - Napisati funkciju *int izracunaj2*(*int n*, *int a*, *int b*) za cele brojeve n, a i b izračunava $(–2)^n * (31a–b–4)$.

46. **hipotenuza** - Napisati funkciju *int hipotenuza*(*int x*, *inty*) koji za date vrednosti kateta x i y izračunava ceo deo hipotenuze z. Formula za izračunavanje hipotenuze z kada su zadate katete x i y glasi $z = sqrt(x^2 + y^2)$. Za računanje celog dela od z treba iskoristiti činjenicu da je to najveći broj čiji kvadrat nije veći od z^2.