Zadaci sa dostupnim rešenjem

1. Napisati potprogram u asemblerskom jeziku kojim se pravi 32-bitna maska:
    a. *unsigned int maska(unsigned int n, unsigned int v)*
    b. Funkcija vraća 32-bitnu vrednost čiji je *n*-ti bit postavljen na vrednost parametra *v* (0 ili 1), dok su svi ostali bitovi postavljeni na suprotnu vrednost od *v*
    c. Ukoliko je neki od parametara neispravan, vratiti nulu; primer za n = 3, v = 0:
        11111111111111111111111111110111
    d. Pokušati implementaciju na osnovu samo stavki a, b i c, a po potrebi pogledati sledeće tačke kao pomoć:
        - Nesipravne vrednosti parametara – za *n*, nevalidna je svaka vrednost veća 31, jer je indeks poslednjeg bita u masci upravo 31; za *v*, nevalidne su vrednosti veće od 1
        - Treba ispitati vrednost ovih parametara i, u slučaju neispravnosti, skočiti na labelu koja rezultat postavlja na nulu
        - U suprotnom, treba postaviti vrednost *v* na odgovarajuću bit poziciju; ukoliko je *v = 1*, ovo se može postići postavljanjem jedinice na bit najmanje težine rezultujućeg registra, i šiftovanjem ovog registra (a time i jedinice) za *n* mesta u levo
        - U slučaju da je *v = 0*, može se prvo jedinica postaviti na *n*-to mesto u registru, pa se onda mogu invertovati bitovi i time dobiti nula na odgovarajućoj poziciji
    e. **Napomena**: rešenje zadatka forsira bit operacije za određene korake zadatka jer je to bila tematika ovih vežbi; nije neophodno koristiti ove operacije u toj meri, i sasvim je korektno da pri izradi zadatka koristite sve što je viđeno na prethodnim terminima vežbi umesto bit operacija, tamo gde je to moguće.
2. Uraditi zadatak iznad, samo u dvostrukoj preciznosti:
    a. *unsigned long long maska(unsigned int n, unsigned int v)*
    b. Sve je isto kao u prethodnom zadatku, samo što ovog puta funkcija vraća 64-bitnu masku
    c. Pokušati implementaciju na osnovu stavki a i b, i prethodnog zadatka, a po potrebi pogledati sledeće tačke kao pomoć:
        Promena za neispravne vrednosti u odnosu na jednostruku preciznost jeste ta što *n* sada može uzeti vrednosti od 0 do 63
        Rešenje se ovde vraća kroz par registara (edx:eax); ukoliko je *n* veće od 31, jedinica preskače u registar edx; treba smestiti jedinicu u eax i šiftovati ovaj registar ukoliko je *n* u opsegu od 0 do 31; ako je veće od 31, treba jedinicu smestiti u edx i šiftovati ovaj registar za vrednost *n-32*; priloženo rešenje koristi nešto drugačiji pristup – šiftuje eax za ostatak pri deljenju *n* sa 32, pa ukoliko je *n* veće od 31 na kraju se zamene vrednosti edx i eax; ovaj deo se može realizovati i korišćenjem šiftovanja u dvostrukoj preciznosti, za šta je kod dostupan u praktikumu na strani 45
3. Napisati potprogram u asemblerskom jeziku kojim se određuje paritet za 15-bitnu vrednost:
    a. *int SetParity(unsigned short int* vrednost)*
    b. Program vraća vrednost bita pariteta (1 ili 0)

   c. Program prima 16-bitnu vrednost po adresi; njen najviši bit je rezervisan za bit parnosti, i potprogram treba da ovaj bit postavi na odgovarajuću vrednost; ostalih 15 bitova predstavljaju bitove realnog podatka za koji se određuje paritet
4. Napisati potprogram u asemblerskom jeziku koji će odrediti horizontalni paritet za niz 15-bitnih brojeva:
   a. *int SetParityArray(unsigned short int* niz, int n)*
   b. Niz sadrži 16-bitne elemente gde je najviši bit svakog elementa rezervisan za bit parnosti
   c. Potprogram treba da poziva potprogram napisan u prethodnom zadatku kako bi postavio najviši bit za svaki element niza i kako bi, kroz svaki od poziva, dobio informaciju o paritetu svakog od elemenata
   d. Povratna vrednost potprograma jeste broj elemenata sa jedinicom na mestu bita pariteta.
   e. Da biste pozivali jedan potprogram iz drugog, u istom fajlu napišite oba potprograma, prvo SetParity (isti potprogram iz prethodnog zadatka), pa ispod njega SetParityArray

Napomena: C programi za ove zadatke dostupni su redom u direktorijumima z1, z2, z3 i z4. Od njih, samo z4 nema skriptu za automatsko testiranje.

Zadaci sa rešenjima dostupnim u praktikumu

1. Napisati potprogram u asemblerskom jeziku koji implementira množenje sabiranjem i pomeranjem:
   a. C program za ovaj zadatak u dvostrukoj preciznosti, kao i potpis funkcije, dostupni su u .c fajlu u direktorijumu z5_mnozenje
   b. Objašnjenje logike, kao i deo rešenja (deo sa glavnom logikom), dostupni su u praktikumu u glavi 7.3.
2. Napisati potprogram u asemblerskom jeziku koji implementira deljenje oduzimanjem i pomeranjem:
   a. C program za ovaj zadatak u dvostrukoj preciznosti, kao i potpis funkcije, dosupni su u .c fajlu u direktorijumu z6_deljenje
   b. Objašnjenje logike, kao i deo rešenja, dostupni su u praktikumu u glavi 7.4.

Zadaci bez dostupnih rešenja

1. Napisati potprogram u asemblerskom jeziku koji proverava paritet za 15-bitnu vrednost:
   a. *int CheckParity(unsigned short int* vrednost)*
   b. Potprogram prima 16-bitni parametar kod koga je najviši bit već postavljen na bit pariteta
   c. Potprogram vraća 1 ukoliko je bit pariteta dobro postavljen, a 0 ukoliko nije
2. Napisati potprogram u asemblerskom jeziku koji proverava paritet za niz 15-bitnih vrednosti:
   a. *int CheckParityArray(unsigned short int* niz, int n)*
   b. Potprogram prima niz 16-bitnih elemenata kod kojih je najviši bit već postavljen na bit pariteta
   c. Potprogram treba da poziva potprogram iz prethodnog zadatka da proveri paritet svakog od elemenata (pročitati stavku *e* kod 4. zadatka u spisku zadataka za koje postoje dostupna rešenja)

d. Potprogram vraća broj elemenata niza kod kojih je bit pariteta pogrešno postavljen

Dodatni zadaci za vežbu:

- Potprogram koji prebrojava jedinice, odnosno nule, u 64-bitnom broju
- Potprogram koji postavlja proizvoljni bit podatka na 1 ili 0 (uraditi i za jednostruku i za dvostruku preciznost)