



Ime i prezime: _____

Broj indeksa: _____

UPUTSTVO ZA RAD

- Popunite svoje podatke na papiru i označite zadatke koje želite da vam budu pregledani u tabeli. Na desktopu napravite folder oblika **arh_kol_ImePrezime_mrGGBBB**.
- Kolokvijum se sastoji od 2 zadataka. Prvi zadatak nosi maksimalno 20 poena, a drugi 30. Broj poena na kolokvijumu dobija se kada se broj osvojenih poena podeli sa 2, i iznosi maksimalno 25 poena.
- Kvalifikacioni zadatak se ne boduje. U slučaju da student ne ume da uradi kvalifikacioni zadatak, rad neće biti dalje pregledan i student automatski ima 0 poena na kolokvijumu.
- Svaki zadatak ima više opcija koje se različito boduju. **Od opcija koje su označene istim slovom, boduje se samo jedna opcija.**
- U slučaju da student uradi više opcija istog zadatka, moli se da obriše opcije **od manje poena**. Npr. student je uradio zadatak 2 opcije a), a*), a**), i sve opcije rade, potrebno je obrisati opcije a) i a*), u slučaju da to ne uradi, nakon što se asistent uveri da je nastupila ta situacija, student će dobiti onoliko bodova koliko nosi najmanja opcija tj. a).
- Student može imati najviše jednu opciju koja radi, i najviše jednu koja ne radi. Opcija koja ne radi mora biti iznad opcije koja radi. Npr. student je uradio zadatak 2, opcije a*) i a***). Opcija a*) radi, a a***) ne radi.
- Ukoliko student uradi više opcija, onog trenutka kada asistent pregleda opciju jednog zadatka koja ne radi, asistent neće pregledati dalje više opcije, ako takve postoje.

Deklaracija akademskog integriteta

Potpisivanjem ovog dela, garantujem svojom čašću da su odgovori na ovom kolokvijumu moj rad bez pomoći drugih osoba i upotrebe nedozvoljenih materijala.

Potpis studenta:

- **(0 poena) Kvalifikacioni zadatak:** Koristeći instrukcije za rad sa registrima opšte namene arhitekture Intel64, napraviti funkciju u assembleru koja će računati sledeći izraz: $x * y - x/2 + 5$. Funkcija treba da bude pozivana iz main.c i ima potpis `int racunajIzraz(int x, int y)`. U main.c se sa standardnog ulaza učitavaju dva cela broja, a na standardni izlaz se štampa rezultat rada funkcije `racunajIzraz` kojoj su ta dva broja prosledjena kao argumenti.

1. Koristeći instrukcije za rad sa registrima opšte namene arhitekture Intel64:

- a) Izračunati nzd dva broja iterativno. Potpis funkcije je `int nzd(int a, int b)`.
- a*) Izračunati nzd dva broja rekursivno koristeći Euklidov algoritam. Potpis funkcije je `int nzd(int a, int b)`.
- b) Koristeći neku od prethodne dve opcije implementirati funkciju koja pronalazi nzd brojeva iz datog niza. Potpis funkcije je `int nzd(int n, int* a)`, gde je n dužina niza.
- c) Odrediti da li jednačina $a_1x_1 + a_2x_2 + \dots + a_nx_n = b$ ima rešenja. Potpis funkcije je `int imaResenja(int n, int* a, int b)` i ona vraća 1 ako jednačina ima rešenja i 0 ako nema. n je dimenzija niza, a u nizu a su smešteni koeficijenti.
- c*) Odrediti da li jednačina $a_1x_1 + a_2x_2 = b$ ima rešenja. Potpis funkcije je `int imaResenja2(int a1, int a2, int b)`. Funkcija vraća 1 ako jednačina ima rešenja i 0 ako nema.

Napisati main.c, unutar koga prvo treba da se učitava dužina niza, a potom i niz celih brojeva. Posle toga se učitava broj b . Kao rezultat na standardni izlaz treba da se ispiše nzd tih brojeva, i to sa predznakom "+" ako `imaResenja(n, a, b)` vrati 1, a u suprotnom sa predznakom "-" ako `imaResenja(n, a, b)` vrati 0. Ako student radi varijantu bez niza, može pretpostaviti da kao ulaz ima dva broja x i y , i dati broj b . Na standardni izlaz treba ispisati `nzd(x, y)`, sa predznakom "+" ako `imaResenja2(x, y, b)` vrati 1, a u suprotnom sa predznakom "-", ako `imaResenja2(x, y, b)` vrati 0. Izvršiti provere da li je dužina niza ceo broj, da li je memorija za niz dobro alocirana, itd. koristeći `assert`. **Sve funkcije iz assemblera implementirati u jednom fajlu 1.s. U main.c jedina spoljna funkcija treba da bude ona iz dela pod c), odnosno ako student radi varijantu bez niza, onda ona koja vraća nzd dva broja. Poželjno je napisati Makefile.**

2. Koristeći SSE instrukcije arhitekture Intel64:

a) Izračunati kosinus ugla između dva vektora na klasičan način. Potpis funkcije je `void Kosinus(int n, float* a, float* b, float* rez)`.

a*) Izračunati kosinus ugla između dva vektora radeći paralelno. Potpis funkcije je `void Kosinus(int n, float* a, float* b, float* rez)`.

a)** Dat je niz vektora spakovan u matricu i dati vektor x . Od svih uglova koje ti vektori zaklapaju sa x naći najmanji. Uglove računati na klasičan način kao i minimum. Potpis f-je je `int MinUgao(int n, int m, float** a, float* x)`. Smatrati da je ugao između 0 i 90 stepeni. Vratiti poziciju vektora u matrici koji zaklapa najmanji ugao sa x .

a*)** Sve isto kao pod (a**) osim što treba koristiti paralelne instrukcije da se izračunaju kosinusi uglova. Minimum računati na klasičan način.

Kosinus ugla između dva vektora $a = (a_1, a_2, \dots, a_n)$ i $b = (b_1, b_2, \dots, b_n)$ računa se po formuli:

$$\cos(a, b) = \frac{a_1 b_1 + a_2 b_2 + \dots + a_n b_n}{\sqrt{a_1^2 + a_2^2 + \dots + a_n^2} * \sqrt{b_1^2 + b_2^2 + \dots + b_n^2}}$$

Napisati `main.c` i odatle pozivati odgovarajuću funkciju. U slučaju da student radi sa matricom, najpre se učitavaju dimenzije matrice, potom sama matrica, a zatim vektor x . Na standardni izlaz treba ispisati poziciju vektora u matrici (tj. red u kome se taj vektor nalazi) koji sa x zaklapa najmanji ugao. Ukoliko student radi neku od prve dve opcije, sa standardnog ulaza se učitava najpre dimenzija vektora, a potom koordinate dva vektora. Na standardni izlaz se ispisuje kosinus ugla koji ta dva vektora zaklapaju. Izvršiti provere da li su dimenzije pozitivni brojevi, da li je alokacija memorije uspešna, itd. koristeći `assert`. **Program nazvati 2.s. Poželjno je napisati Makefile.**

Primer ulaza za koji je izlaz 1: `n=4 m=3`

matrica = `[-0.7 12.6 -7.4, 13.16 15.7 5.04, 11.75 4 6.46, 1 1 -0.94]`, `x=[1.3 2.19 3.57]`

Kosinusi uglova su redom: 0.00415, 0.75104, 0.76753, 0.01802.

Lista zadataka									
Broj zadatka i opcija	1.a	1.a*	1.b	1.c	1.c*	2.a	2.a*	2.a**	2.a***
Broj max bodova	5	10	7	3	3	10	18	24	30
Da se pregleda									

Srećan rad!

DOZVOLJENA LITERATURA NA KOLOKVIJUMU (SSE)

Pre rada sa SSE instrukcijama proverava se da li procesor podržava ove instrukcije. To se može uraditi na sledeći način:

```
mov rax, 1
cpuid
test rdx, 0x2000000
jz not_supported
```

gde se nakon labele not_supported navode instrukcije za završetak programa.

Zatim se čuva sadržaj SSE registara i registara koprocesora:

```
mov rdx, rsp
and rsp, 0xffffffffffff0
sub rsp, 512
fxsave [rsp]
```

Na kraju rada sa ovim registrima se vraća njihov prvobitni sadržaj:

```
fxrstor [rsp]
mov rsp, rdx
```

- FXSAVE - smešta registre numeričkog koprocesora i sadržaje xmm registara u memoriju
- FXRSTOR - upisuje iz memorije u registre numeričkog koprocesora i u xmm registre sačuvane vrednost

Instrukcije transfera:

Ove instrukcije vrše transfer između dva xmm registra ili xmm registra i memorije.

- MOVAPS - smešta poravnata pakovana 4 realna broja jednostruke preciznosti u lokaciju zadatu prvim operandom
- MOVUPS - smešta neporavnata pakovana 4 realna broja jednostruke preciznosti u odredišni operand
- MOVSS - smešta jedan realan broj jednostruke preciznosti u odredišni operand
- MOVLPS - smešta dva realna broja sa niže adrese u memoriju ili iz memorije u niži deo registra, preostali deo registra ostaje nepromenjen
- MOVHPS - smešta dva realna broja sa više adrese u memoriju ili iz memorije u viši deo registra, preostali deo registra ostaje nepromenjen
- MOVLHPS - smešta niži deo registra u viši deo, niži deo ostaje nepromenjen
- MOVHLPS - smešta viši deo registra u niži deo, viši deo ostaje nepromenjen
- MOVMSKPS - smešta po jedan najviši bit sva četiri podatka u registre opšte namene

Aritmetičke instrukcije:

- ADDPS - sabira paralelno po četiri podatka
- SUBPS - oduzima paralelno po četiri podatka
- ADDSS - sabira samo podatak na najnižoj adresi
- SUBSS - oduzima samo podatak na najnižoj adresi
- MULPS - množi paralelno četiri podatka
- MULSS - množi samo podatke na najnižoj adresi
- DIVPS - deli paralelno četiri podatka
- DIVSS - deli samo podatke na najnižoj adresi
- RCPPS - računa paralelno recipročne vrednosti podataka
- RCPSS - računa recipročnu vrednost podatka na najnižoj adresi
- SQRTPS - računa paralelno kvadratni koren

- SQRTSS - računa kvadratni koren podatka na najnižoj adresi
- RSQRTPS - računa recipročnu vrednost kvadratnog korena paralelno
- RSQRTPS - računa recipročnu vrednost kvadratnog korena podatka na najnižoj adresi
- MAXPS - računa maksimume paralelno
- MAXSS - računa maksimum podatka na najnižoj adresi
- MINPS - računa minimume paralelno
- MINSS - računa minimume podatka na najnižoj adresi

Logičke instrukcije:

- ANDPS - logička konjunkcija
- ANDNPS - logička negacija konjunkcije
- ORPS - logička disjunkcija
- XORPS - logička ekskluzivna disjunkcija

Instrukcije poređenja:

- CMPPS - poredi prema trećem argumentu instrukcije, popunjava svim jedinicama ako su podaci jednaki, u suprotnom popunjava svim nulama, treći argument može biti 0h jednako, 1h manje, 2h manje jednako, 3h neuporedivi, Dh veće ili jednako, Eh veće
- CMPSS - poredi samo podatak na najnižoj adresi prema trećem argumentu instrukcije, popunjava svim jedinicama ako su podaci jednaki, u suprotnom popunjava svim nulama
- COMISS - poredi samo podatak na najnižoj adresi i postavlja EFLAGS registar, tako da se mogu koristiti instrukcije skoka za neoznačene operande, signalizira izuzetak ako je operand NaN

Instrukcije šiftovanja:

- SHUFPS - smešta bilo koja dva od četiri podatka u niži deo odredišnog operanda i bilo koja dva podatka u viši deo odredišnog operanda. Ukoliko su oba operanda isti registar smešta podatke u bilo kom redosledu
- UNPCKHPS - kombinuje po dva podatka sa viših adresa
- UNPCKLPS - kombinuje po dva podatka sa nižih adresa

Instrukcije za konverziju:

- CVTPI2PS - konvertuje cele brojeve u jednostruku preciznost paralelno
- CVTSI2SS - konvertuje ceo broj u jednostruku preciznost
- CVTPS2PI - konvertuje jednostruku preciznost u cele brojeve paralelno
- CVTSS2SI - konvertuje jednostruku preciznost u ceo broj

DOZVOLJENA LITERATURA NA KOLOKVIJUMU (INSTRUKCIJE ZA RAD SA REGISTRIMA OPŠTE NAMENE)

Rad sa stekom: push, pop

Instrukcije transfera:

- MOV op1, op2 - premeštanje (op1 = op2)
- MOVZX op1, op2 - u prvi operand se smešta vrednost drugog operanda proširena nulama
- MOVSX op1, op2 - u prvi operand se smešta vrednost drugog operanda proširena u skladu sa znakom
- LEA op1, op2 - učitavanje adrese drugog operanda u prvi operand, pri čemu je drugi operand memorijski operand

Aritmetičke instrukcije:

- CDQE - označeno proširivanje eax na rax
- CDQ - označeno proširivanje eax na edx:eax
- CQO - označeno proširivanje rax na rdx:rax

Standardne: ADD, SUB, DIV, IDIV (označeno deljenje), MUL, IMUL(označeno množenje), NEG(promena znaka), INC, DEC

Standardne logičke: AND, OR, XOR, NOT, SHL, SHR(logičko šiftovanje u desno), SAR(aritmetičko šiftovanje u desno).

Poređenje: CMP, TEST(testiranje bitova)

Instrukcije kontrole toka:

- JMP op - bezuslovni skok na adresu op (memorijski operand)
- CALL op - bezuslovni skok uz pamćenje povratne adrese na steku.
- RET - skida sa steka adresu i skače na tu adresu.
- JZ op - skače ako je rezultat prethodne instrukcije nula.
- JE op - skače ako je rezultat prethodnog poređenja jednakost (ekvivalentno sa JZ)
- JNZ op - skače ako je rezultat prethodne operacije različit od nule
- JNE op - skače ako je rezultat prethodnog poređenja različitost (ekvivalentno sa JNZ)
- JA op - skače ako je rezultat prethodnog poređenja veće (neoznačeni brojevi)
- JB op - skače ako je rezultat prethodnog poređenja manje (neoznačeni brojevi)
- JAE op - skače ako je rezultat prethodnog poređenja veće ili jednako (neoznačeni brojevi)
- JBE op - skače ako je rezultat prethodnog poređenja manje ili jednako (neoznačeni brojevi)
- JG op - skače ako je rezultat prethodnog poređenja veće (označeni brojevi)
- JL op - skače ako je rezultat prethodnog poređenja manje (označeni brojevi)
- JGE op - skače ako je rezultat prethodnog poređenja veće ili jednako (označeni brojevi)
- JLE op - skače ako je rezultat prethodnog poređenja manje ili jednako (označeni brojevi)

Slično, postoje i negacije gornjih instrukcija uslovnog skoka: JNA, JNB, JNAE, JNBE, JNG, JNL, JNGE, JNLE