Arhitektura računara

Projektni zadaci – arhitektura instrukcijskog skupa (grupa 1)

Zadatak 1.1 – Simulator arhitekture instrukcijskog skupa

(12) U proizvoljnom programskom jeziku napisati simulator sopstvene arhitekture instrukcionog skupa sa bar 4 registra opšte namjene dužine 64 bita (dozvoljeno je koristiti tip podataka dužine 8 bajta, npr. *long long* ili *uint64_t* za registre). Simulator treba da funkcioniše kao interpreter. Treba biti omogućeno da se izvorni asemblerski kod učita iz fajla. Pravilno izvršiti potrebnu leksičku, sintaksičku i semantičku analizu koda. Instrukcijski skup simulirane mašine (gosta) mora da obuhvata:

- osnovne aritmetičke operacije (ADD, SUB, MUL, DIV)
- osnovne bitske logičke operacije (AND, OR, NOT, XOR)
- instrukciju za pomjeranje podataka između registara (MOV)
- instrukciju za unos podataka sa standardnog ulaza (slično odgovarajućem sistemskom pozivu)
- instrukciju za ispis podataka na standardni izlaz (slično odgovarajućem sistemskom pozivu)

Implementirati jednostavnu *single-step debugging* podršku. Omogućiti izvršavanje i pregled vrijednosti svih registara i specificiranih memorijskih adresa na postavljenim zaustavnim tačkama u asemblerskom kodu tokom izvršavanja. U ovom režimu treba biti omogućen prelaz na sljedeću instrukciju (NEXT ili STEP konzolne komande) i prelaz na sljedeću zaustavnu tačku (CONTINUE).

(4) Implementirati rad sa memorijom. Simulirana mašina (gost) treba da ima 64-bitni adresni prostor. Omogućiti direktno i indirektno adresiranje. Sadržaj svake memorijske adrese treba biti dužine 1 bajt. Pravilno omogućiti da se adrese mogu navesti kao brojevi. Omogućiti pristup svim adresama iz adresnog prostora, uključujući i upis i čitanje, upotrebom odgovarajućih instrukcija (*MOV* ili *LOAD/STORE*).

(4) Implementirati instrukcije potrebne za bezuslovno i uslovno grananje (JMP, CMP, JE, JNE, JGE, JL).

(5) Umjesto direktnog interpretiranja stringova, omogućiti da se asemblerske instrukcije mogu prevesti u mašinski kod gosta (bajtkod) i smjestiti u adresni prostor gosta (analogno *code* ili *text* segmentu). Instrukcije, dakle, treba da se interpretiraju i izvršavaju iz memorije gosta. Definisati i iskoristiti registar koji će da služi kao programski brojač (pokazivač na instrukciju). Konstruisati primjer asemblerskog koda za gosta koji je samomodifikujući (nakon prevođenja u mašinski kod gosta) i u kom se modifikovani dio koda izvršava i prije i poslije relevantne modifikacije.

Obezbijediti nekoliko jediničnih testova kojima se demonstriraju funkcionalnosti iz stavki u specifikaciji projektnog zadatka.

Sve detalje koji nisu eksplicitno navedeni implementirati na proizvoljan način. Pridržavati se:

- principa objektno-orijentisanog programiranja i SOLID principa
- principa pisanja čistog koda i pravilnog imenovanja varijabli, funkcija, klasa i metoda
- konvencija za korišteni programski jezik

Zadatak 1.2 – Asemblerski program za obradu podataka

(15) Napisati asemblerski program za obradu podataka. Algoritam odabrati na stranici kursa. Program treba da, kao argumente komandne linije, prihvata putanju do ulaznog fajla, putanju do izlaznog fajla, kao i vrijednosti parametara algoritma. Obezbijediti smislene podrazumijevane vrijednosti za sve argumente komandne linije. Program treba da dinamički alocira stranice potrebne za podatke koji se obrađuju.

(5) Optimizovati program uvođenjem SSE ili AVX paralelizma, te dokumentovati ubrzanje.

(5) Napisati isti program u C ili C++ programskom jeziku i isprobati različite nivoe kompajlerskih optimizacija. Pri tome, pridržavati se osnovnih principa pisanja efiksanog koda. Uporediti performanse sa prethodnim implementacijama i dokumentovati rezultate.

Zadatak 1.3 – Kernel operativnog sistema

Realizovati kernel operativnog sistema koji koristi proizvoljan *bootloader* (npr. GRUB).

(12) Implementirati jednostavan kalkulator sa podrškom za sabiranje, oduzimanje, množenje i dijeljenje 32-bitnih cijelih brojeva koje korisnik unosi putem tastature.

(4) Implementirati prikaz sistemskog vremena ili prikaz neke informacije o hardveru računara.

(4) Implementirati promjenu boje teksta pri svakom otkucaju sistemskog tajmera.

(5) Implementirati prelazak u 64-bitni režim rada**.** Pri tome, straničenje treba biti podešeno tako da u memoriji istovremeno bude prisutna bar jedna 1GB stranica, bar jedna 2MB stranica i bar jedna 4KB stranica. Pri tome, sve ostale stavke iz zadatka treba da budu realizovane u 64-bitnom režimu rada.