# Izvedba Jaguar biblioteke na FRISC-V procesoru

**Bevanda, Daria**

SVEUČILIŠTE U ZAGREBU
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

ZAVRŠNI RAD br. 1259

# IZVEDBA JAGUAR BIBLIOTEKE NA FRISC-V PROCESORU

Daria Bevanda

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

ZAVRŠNI RAD br. 1259

# IZVEDBA JAGUAR BIBLIOTEKE NA FRISC-V PROCESORU

Daria Bevanda

Zagreb, lipanj 2024.

**SVEUČILIŠTE U ZAGREBU**
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

Zagreb, 4. ožujka 2024.

# ZAVRŠNI ZADATAK br. 1259

| | |
|---|---|
| Pristupnica: | **Daria Bevanda (0036538432)** |
| Studij: | Elektrotehnika i informacijska tehnologija i Računarstvo |
| Modul: | Računarstvo |
| Mentor: | prof. dr. sc. Mario Kovač |

Zadatak: **Izvedba Jaguar biblioteke na FRISC-V procesoru**

Opis zadatka:

Proučiti RISC-V arhitekturu procesora i Jaguar biblioteku za kompresiju i dekompresiju slika. Napisati, prevesti i ispitati programsko rješenje za prikaz JPEG slike na LCD ekranu ulazno/izlazne FRISC-V pločice korištenjem Jaguar biblioteke.

Rok za predaju rada: 14. lipnja 2024.

# Sadržaj

# 1. Uvod

U suvremenom tehnološkom okruženju, digitalna obrada slika postaje sve važnija u mnogim područjima, uključujući nadzor, prepoznavanje objekata i medicinsku dijagnostiku. Ovaj rad bavi se proučavanjem RISC-V arhitekture procesora i Jaguar biblioteke za manipulaciju slikama, kako bi se omogućila efikasna obrada slika izravno na procesoru. Cilj je istražiti i demonstrirati kako se pomoću otvorenih standarda poput RISC-V i specijaliziranih biblioteka poput Jaguar može optimizirati proces obrade slika u realnom vremenu, smanjujući potrebu za vanjskom obradom i složenim hardverskim rješenjima.

RISC-V arhitektura je otvoreni standard instrukcijskog seta (ISA) koji nudi jednostavnost, modularnost i prilagodljivost, čineći ga pogodnim za različite primjene. Jaguar biblioteka, napisana u programskom jeziku C, pruža alate za dekodiranje i kodiranje slika, omogućujući brzu i efikasnu obradu. Kroz ovaj rad istražit ćemo tehničku izvedbu povezivanja računala i procesora, prijenos datoteka, manipulaciju slikama pomoću Jaguar biblioteke te prikaz slika na OLED ekranu.

Ovaj projekt također naglašava izazove i rješenja pri implementaciji sustava koji zahtijevaju visoku učinkovitost i nisku potrošnju energije. Posebna pažnja posvećena je praktičnoj implementaciji i testiranju funkcionalnosti na ARM procesoru, s ciljem da se stečeno znanje prenese na FRISC-V procesor u budućnosti. Kroz rad ćemo prikazati važnost otvorenih standarda i optimiziranih biblioteka u modernim sustavima obrade podataka.

# 2. Teorijska podloga

U svrhu uspješnog razumijevanja zadatka, potrebno je proučiti arhitekturu samog RISC-V procesora te Jaguar biblioteku korištenu za manipulaciju slikama.

**RISC-V** je otvoreni standard skupa naredaba (ISA) koji je temeljen na specifikaciji naredaba procesora RISC (*Reduced Instruction Set Computer*). Glavna karakteristika RISC-V ISA je njena jednostavnost i modularnost, što je čini jednostavnom za implementaciju i prilagodljivom različitim potrebama korisnika. Osnovni skup instrukcija RISC-V obuhvaća uobičajene operacije kao što su aritmetičke i logičke operacije te upravljanje i prijenos podataka. Osnovnu funkcionalnost može se proširiti dodatnim modulima koji pružaju specifične mogućnosti poput množenja i dijeljenja (M proširenje), atomskih instrukcija za paralelno programiranje (A proširenje), rad sa brojevima u pokretnom zarezu (F i D proširenja), vektorskih instrukcija za visoki nivo paralelizma (V proširenje) i komprimiranih instrukcija koje smanjuju veličinu programa (C proširenje). Zahvaljujući malom skupu osnovnih naredaba i mogućnosti različitih širina adresa, RISC-V arhitektura je vrlo prilagodljiva specifičnim potrebama različitih aplikacija [1].

**Jaguar biblioteka** je razvijena s ciljem manipulacije slikama i njihovim formatima. Njene osnovne funkcije uključuju dekodiranje i kodiranje slika. Dekodiranjem se ulazna slika iz formata JPEG ili JPG pretvara u PGM (*Portable Gray Map*) format, koji sadrži jednu komponentu boje, ili u PPM (*Portable Pix Map*) format, koji sadrži tri komponente boje. Kodiranjem se, pak, slike iz PGM ili PPM formata pretvaraju natrag u JPEG ili JPG format, što uključuje kompresiju slike kako bi se smanjila veličina dokumenta bez značajnog gubitka kvalitete.

Jaguar biblioteka je napisana u programskom jeziku C, što je čini izuzetno efikasnom i pogodnom za pokretanje na različitim platformama, uključujući ARM i RISC-V procesore. Zahvaljujući optimizaciji za ove arhitekture, Jaguar biblioteka omogućava brzu i efikasnu obradu slika, što je ključno za aplikacije koje zahtijevaju brzu obradu podataka i nisku potrošnju energije.

# 3. Tehnička izvedba

## 3.1. Komunikacija računala i procesora te prijenos datoteke

Kako je zadatak provesti kompresiju slike na samom procesoru FRISC-V te prikaz na OLED ekranu koji se nalazi na perifernoj pločici BaseBoard, prvotno je bilo potrebno uspostaviti komunikaciju između računala na kojemu se nalazi slika te pločice, odnosno procesora kako bi se mogao odviti transfer slike na procesor.

Za početak korištena je pločica PYNQ-Z1 na kojoj se nalazi ARM procesor radi jednostavnosti arhitekture samog procesora te lakše implementacije tražene funkcionalnosti [4]. Serijskom vezom se uspostavlja komunikacijski kanal u oba smjera, tako da se pločica s procesorom direktno, žicom, povezuje na prethodno određeni priključak (COM6). Na računalu se nalazi programski kod kojim se šalje veličina slike koja je trenutno u JPEG (JPG) formatu, s jednom komponentom boje, te njen sadržaj. Ovaj kod izveden je u *Python* programskom jeziku radi jednostavnosti sintakse i niske zahtjevnosti funkcionalnosti koju računalo mora provesti.

Nakon slanja veličine i sadržaja slike na COM6 serijski priključak, program pisan u jeziku C aktivno čita sadržaj porta te sprema dio po dio dok ne pročita onoliko bajtova podataka kolika je veličina poslane datoteke. U svrhu utvrđivanja ispravnosti provedene komunikacija, procesor, nakon što pročita sav sadržaj s priključka koji mu je računalo poslalo, na isti COM6 priključak vraća i šalje natrag primljeni sadržaj i veličinu datoteke istim komunikacijskim kanalom te računalo čita podatke i stvara novu sliku koja bi, ako se komunikacija provela bez ikakve greške, trebala biti jednaka početnoj.

Iako za komunikaciju nije implementirana nikakva vrsta rukovanja, proces slanja i primanja sadržaja slike može se pratiti u ispis programa koji se izvršava na računalu. Tamo korisnik može vidjeti, tijekom pokrenute komunikacije, koji iznosi veličine datoteke računalo šalje, bajtove poslanog sadržaja, te napredak pri slanju podataka (broj poslanih podataka kroz ukupan broj podataka za slanje). Iste te podatke računalo ispisuje pri prihvaćanju poslanih podataka od strane procesora te ispisuje informacije o veličini nove datoteke te napredak primljenih podataka.

Nakon što se komunikacija uspješno uspostavila između računala i samog procesora, slijedi dekodiranje zaprimljene datoteke iz JPEG formata u PGM format pomoću Jaguar biblioteke. Dekodiranje se provodi isključivo iz JPG u PGM format, koji se sastoji samo od nijansi sive boje, nad slikom dimenzija 96x64, a to ograničenje određuje OLED ekran na kojemu će se slika morati prikazati.

### 3.2. Primjena Jaguar biblioteke radi manipulacije slike

Postupak dekodiranja sastoji se od nekoliko koraka; prvo se provodi inicijalizacija Jaguar biblioteke gdje se kao rezultat dohvaćaju osnovni podatci o slici koja ulazi u proces. Podatci o slici koji se prikupljaju u tom procesu su širina, visina, broj komponenti boje (koji je za potrebe ovog završnog rada isključivo 1) te preciznost koja može biti 8 ili 12 bitova (u ovom slučaju isključivo 8 bitova). Nakon uspješne inicijalizacije, slijedi konkretan postupak dekodiranja. U prvom koraku se kreira spremnik podataka odgovarajuće veličine koji će primiti dekomprimirane podatke. Veličina spremnika izračunava se na temelju širine, visine i broja komponenti boje slike (u ovom slučaju to su vrijednosti 96, 64 te 1). Zatim se poziva funkcija *jaguar_decode_8bit*, koja je zadužena za dekodiranje JPEG podataka i njihovo spremanje u spremnik. Unutar funkcije *jaguar_decode_8bit* provjerava se ispravnost parametara i alociranog spremnika, a zatim se vrši dekodiranje koristeći *Huffmanovo* dekodiranje putem funkcije scanDecode_8bit. Ova funkcija dekodira komprimirane JPEG podatke u sirove, neobrađene piksele koristeći unaprijed definirane *Huffmanove* tablice. Nakon što se slika uspješno dekodira, prije samoga spremanja, provodi se izmjena vrijednosti bajtova kako bi poprimali vrijednosti isključivo bijele, odnosno crne boje. Konverzija boja, odnosno vrijednosti bajtova slike, provodi se na način da se uspoređuje vrijednost bajta te ga se pretvara u boju kojoj je po svojoj vrijednosti bliži. Da bi se mogla odrediti „blizina“ vrijednost, postavlja se granica skale prema kojoj će se moći odrediti je li boja bliža crnoj (0x00) ili bijeloj (0xFF). Uz činjenicu da se boje nalaze u rasponu od 0 do 255 (u bajtovima su boje zapisane kao dvoznamenkasti heksadekadski brojevi) granica pretvorbe može biti ili 127 ili 128, a odluka za bilo koju od tih vrijednosti ne utječe na uspješnosti

provedbe izmjene. U kodu je kao granica postavljena vrijednost 127; dakle bajtovi čije vrijednosti su manje od iznosa 127 postavljaju se na vrijednost 0, dok se bajtovi veći ili jednaki od 127 postavljaju na vrijednost 255. Sva navedena funkcionalnost vezana uz pretvorbu boja implementirana je u funkciji *convert_to_black_and_white.*

```
void convert_to_black_and_white(unsigned char *buffer, size_t buffer_size) {
        int threshold = 127;
        for (size_t i = 0; i < buffer_size; i++) {
                buffer[i] = (buffer[i] <= threshold) ? 0 : 255;
        }
}
```

Kod 3.2.1. - Implementacija pretvorbe slike u crno-bijelu

Nakon što je slika promijenjena tako da bude isključivo crno-bijela, sada ju je potrebno spremiti, no kako se kompletan kod namijenjen za dekompresiju i konverziju boja izvršava na procesoru, koji nema sustav datoteka, nije moguće koristiti postojeće funkcije spremanja nove slike koje Jaguar biblioteka posjeduje. Za savladavanje ovog problema, implementirana je nova funkcija *pgmSave_8b_mem* po uzoru na postojeću funkciju *pgmSave_8b* koja se nalazi u biblioteci, s osnovnom razlikom da se ne stvara nova datoteka formata PGM, već se u radnu memoriju (na gomilu) sprema izmijenjeni sadržaj slike s nadodanim sučeljem koje je potrebno kako bi se slika uspješno spremila na računalu u ispravnom formatu. Važna napomena jest da je, zbog velike količine podataka koji se spremaju u alociranu memoriju te su potrebni kroz cijeli implementirani proces, potrebno povećati standardnu veličinu gomile kako bi bila dovoljno velika da ispuni zahtjeve implementiranoga rješenja.

```
int pgmSave_8b_mem(const unsigned char *buffer, const size_t buffer_size,
                const int width, const int height, const int maxval,
                unsigned char **pgm_buffer, size_t *pgm_size) {

        if (maxval > 255) {
                printf("Error: Maxval bigger than 255! Maxval = %d\n", maxval);
                return 1;
        }

        char header[32];
        int header_size = snprintf(header, sizeof(header), "P5\n%d %d\n%d\n", width,
                        height, maxval);

        // Allocate memory for the PGM data
        *pgm_size = header_size + buffer_size;
        *pgm_buffer = (unsigned char *) malloc(*pgm_size);
        if (*pgm_buffer == NULL) {
                printf("Memory allocation failed\n");
                return 1;
        }

        // Copy the header and image data into the PGM buffer
        memcpy(*pgm_buffer, header, header_size);
        memcpy(*pgm_buffer + header_size, buffer, buffer_size);

        return 0;
}
```

Kod 3.2.2. – Implementacija spremanja izmijenjene slike u PGM formatu na stog

## 3.3. Prikaz slike na OLED ekranu

Preduvjet za uspješan prikaz dekomprimirane crno-bijele slike na OLED ekranu jest generiranje „bitstream“-a s kojim se po vlastitim potrebama konfigurira blok dizajn FPGA pločice te tako osigurava ispravna komunikaciju procesora te BaseBoard pločice koja na sebi sadrži OLED ekran [2]. Nakon uključivanja ispravnog blok dizajna prevedenog u niz bitova („bitstream“) fokus se prebacuje na samu implementaciju koda koji će uspostavit komunikaciju s ekranom te prenijeti sadržaj slike i prikazati ga. To se postiže GPIO sklopovima koji se prethodno moraju ispravno konfigurirati u bitstreamu. [3] Korištenje OLED ekrana na ispravan način uključuje sljedeće korake:

1. Dovođenje napona
2. Konfiguracija zaslona: Slanje naredbe za gašenje zaslona (0xAE) i drugih specifičnih naredbi za konfiguraciju prikaza.
3. Paljenje zaslona
4. Slanje podataka: Podaci se šalju red po red u internu memoriju OLED ekrana putem SPI sučelja.
5. Gašenje zaslona

Za 4. korak slanja podataka ekranu, potrebno je naglasiti da je bitna struktura podataka koji se šalju za prikaz. U tu svrhu definira se struktura *data_sample* čija svrha je spremanje podataka na način na koji ih ekran može pročitati i prikazati.

```
struct data_sample {
        uint8_t data[SAMPLE_SIZE];
};
```

Kod 3.3.1. - Podatkovna struktura *data_sample*

Dakle, potrebno je pretvoriti podatke slike u listu bajtova za prikaz. U tu svrhu implementirana je funkcija *parse_pgm_data_and_display* čija je funkcionalnost čitanje podataka, preskakanje prva 3 retka koji su sučelje same slike (uvijek 3 jer će se uvijek prikazivati podatci slike u PGM formatu) te prolazak kroz sve podatke i uspoređivanje vrijednosti bajtova s 0 i 1 (odnosno crna ili bijela boja). Nakon konstruiranja podatka koji se može prikazati na ekranu, poziva se funkcija za prikaz te se bajt po bajt šalju ekranu.

```
void parse_pgm_data_and_display(const unsigned char *pgm_data, size_t pgm_data_size) {
    const unsigned char *image_data = pgm_data;
    while (*image_data != '\n')
        image_data++;
    image_data++;
    while (*image_data != '\n')
        image_data++;
    image_data++;
    while (*image_data != '\n')
        image_data++;
    image_data++;

    const int width = 96;
    const int height = 64;
    const int pages = height / 8;

    unsigned char *byte_data = (unsigned char *)malloc(width * height);
    memcpy(byte_data, image_data, width * height);

    struct data_sample sample;
    memset(&sample, 0, sizeof(struct data_sample)); // Clear the sample data

    // Parse the image data into the sample structure
    int data_index = 0;
    for (int page = 0; page < pages; ++page) {
        for (int col = 0; col < width; ++col) {
            unsigned char byte = 0;
            for (int row = 0; row < 8; ++row) {
                int index = (page * 8 + row) * width + col;
                byte |= ((byte_data[index] ? 1 : 0) << row);
            }
            sample.data[data_index++] = byte;
        }
    }

    oled_send_sample(&sample);

    free(byte_data);
}
```

Kod 3.3.2. - Implementacije funkcije za konstrukciju podataka za prikaz

# 4. Testiranje i verifikacija

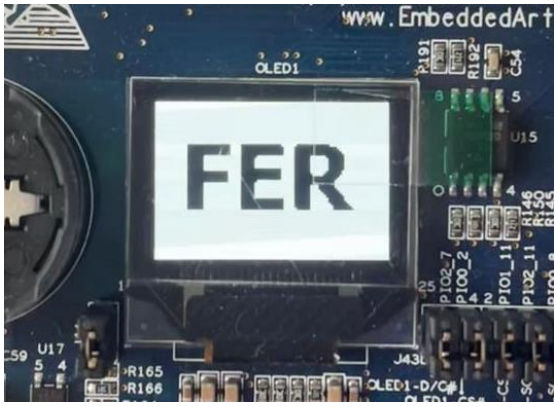
Uspješnost implementacije gore navedenih funkcionalnosti može se pokazati tako da se provede jedan ciklus slanja slike s računala na procesor na kojemu se potom provodi primanje sadržaja slike, pretvaranje slike u prethodno dogovoreni format (PGM) s pomoću biblioteke Jaguar, mijenjanje vrijednosti bajtova tako da poprimaju vrijednosti isključivo crne, odnosno bijele boje te na kraju slanje i prikazivanje izmijenjene slike na OLED ekranu. Kao eksperimentalnu sliku koristi se slika pod nazivom FER_grayscale.jpg.


Slika 4.1. - FER_grayscale.jpg

Prvotno se pločica na kojoj se nalazi procesor mora spojiti na računalo isključivo preko COM6 porta, koji je predefiniran u samoj implementaciji koda koji se pokreče i na računalu i na procesoru. Nakon što se pločica uspješno poveže, bitno je da se prvo pokreće program na *Vitis* platformi koji aktivno sluša događanja na određenom portu te čeka podatke koje šalje računalo, a potom se pokreće kod namijenjen za slanje slike s računala. Procesor će računalu vratiti izmijenjenu sliku te se time može provjeriti sadržaj slike i uvjeriti se da se bajtovi isključivo u vrijednostima bijele i crne boje.


Slika 4.2. - Zaprimljena slika nakon dekompresije i izmjene boja

To je također vidljivo i na samom ekranu koji uspješno prikazuje poslanu sliku.




Slika 4.3. - Prikaz slike na OLED ekranu

Verifikacija se provodi usporedbom prvotne veličine poslane slike te provjerom sadržaja zaprimljene slike, kao i samim vidljivim prikazom te slike na ekranu kojom se može potvrditi da su sve uspostavljene komunikacije uspješne. Prvotna slika je dimenzija 96 x 64 te veličine 1.70 KB (1,743 bajtova), dok je zaprimljena slika jednakih dimenzija, ali veličine 6.01 KB (6,157 bajtova) što je očekivano jer se radi o nekomprimiranom formatu PGM u kojemu se svaki piksel sprema u svojoj punoj veličini.

## 5. Zaključak

Cilj ovog rada bio je uspostaviti komunikaciju između računala, FRISC-V procesora i OLED ekrana na BaseBoard pločici te demonstrirati manipulaciju slikama pomoću Jaguar biblioteke. Iako se proces testiranja i validacije nije uspio izvesti na samom FRISC procesoru, već samo na ARM procesoru, postignuti rezultati pokazuju da je komunikacija između računala i procesora izvediva jednostavnim kodom.

Kroz tehničku izvedbu projekta, uspješno je uspostavljena dvosmjerna komunikacija putem serijske veze (COM6 porta). Implementacija je uključivala prijenos slike u JPEG formatu s računala na procesor, njeno dekodiranje u PGM format pomoću Jaguar biblioteke, te konverziju slike u crno-bijeli format prilagođen za prikaz na OLED ekranu. Za ovu funkcionalnost, razvijene su specifične funkcije za dekodiranje, manipulaciju i spremanje slike u radnu memoriju procesora.

Izazovi su se pojavili prilikom integracije Jaguar biblioteke u okruženje ARM procesora, posebno zbog nemogućnosti korištenja standardnih biblioteka, što je zahtijevalo pronalazak alternativnih rješenja. Na primjer, funkcija za spremanje izmijenjene slike morala je biti prilagođena za rad u ograničenom memorijskom okruženju procesora.

Primjena ove tehnologije može značajno unaprijediti obradu slika u realnom vremenu. Konkretno, sustavi nadzora koji koriste algoritme za prepoznavanje objekata, kao što su automobili u parkiralištima, mogli bi imati velike koristi od ovog pristupa.

## 6. Literatura


[1] Kovač M., Arhitektura računala 1R, predavanja 2023.

[2] Priručnik za rad s OLED ekranom (https://www.codico.com/en/mpattachment/file/download/id/406/)

[3] Grzunov M., Seminarski rad 1, studeni 2022.

[4] Priručnik za rad s Pynq-Z1 pločicom, Digilent (An NI Company) (https://digilent.com/reference/programmable-logic/pynq-z1/reference-manual?redirect=1)

# 7. Sažetak

**Izvedba Jaguar biblioteke na FRISC-V procesoru**

Ovaj završni rad je trebao istražiti mogućnosti primjene RISC-V arhitekture procesora i Jaguar biblioteke za manipulaciju slikama radi postizanja efikasne obrade slika izravno na procesoru. U radu se uspostavila komunikacija između računala, ARM procesora na PYNQ-Z1 pločici i OLED ekrana na BaseBoard pločici te demonstrirala manipulacija slikama pomoću Jaguar biblioteke. Projekt je uključivao prijenos slike s računala na procesor, dekodiranje slike u PGM format, konverziju u crno-bijelu sliku te prikaz slike na OLED ekranu. Implementacija je uspješno dokazala da je moguće ostvariti dvosmjernu komunikaciju i obradu slika koristeći navedene tehnologije. Iako testiranje nije provedeno na FRISC-V procesoru, rezultati su pokazali da je arhitektura pogodna za primjenu u sustavima nadzora i sličnim aplikacijama koje zahtijevaju brzu i efikasnu obradu slika.

Ključne riječi: Jaguar biblioteka, obrada slika, komunikacija računala i procesora, JPEG, PGM, dekompresija slike

# 8. Summary

**Implementation of the Jaguar Library on the FRISC-V Processor**

This final thesis aimed to explore the potential application of the RISC-V processor architecture and the Jaguar library for image manipulation to achieve efficient image processing directly on the processor. In the thesis communication between a computer, an ARM processor on the PYNQ-Z1 bord was established, and an OLED display on the BaseBoard, and image manipulation using the Jaguar library demonstrated. The project involved transferring an image from the computer to the processor, decoding the image to PGM format, converting it to a black-and-white image, and displaying the image on the OLED screen. The implementation successfully demonstrated the feasibility of achieving bidirectional communication and image processing using these technologies. Although testing was not conducted on the FRISC-V processor, the results indicated that the architecture is suitable for applications in surveillance systems and similar areas that require fast and efficient image processing.

Key Words: Jaguar library, image processing, computer-processor communication, JPEG, PGM, image decompression

# 9. Skraćenice

| | | |
|---|---|---|
| ISA | *Instruction Set Architecture* | Arhitektura seta naredaba |
| FPGA | *Field Programmable Gate Arrays* | Pogramirljivo polje logičkih sklopova |
| GPIO | *General Purpose Input/Output* | Ulazno-izlazna jedinica opće namjene |