Melih-Muhammed Ayabaktı

Documentatie CHIP-8

Profesor Coordonator: Cristina Olaru

**Introducere**

În cadrul proiectului de atestat la informatică, am ales să dezvolt un emulator pentru sistemul CHIP-8, folosind limbajul C și biblioteca grafică SDL2. Alegerea acestui subiect reflectă interesul meu pentru programarea de nivel jos, arhitectura calculatoarelor și modul în care sistemele vechi pot fi reproduse și înțelese prin cod modern. Documentația de față urmărește să explice atât principiile teoretice din spatele sistemului CHIP-8, cât și detaliile concrete ale implementării unui emulator complet funcțional.

**Emulatorul CHIP-8 în limbajul C (biblioteca SDL2)**

**O scurta prezentare**

CHIP-8 este un limbaj interpretat creat în 1977 de inginerul Joe Weisbecker pentru microcalculatorul RCA COSMAC VIP. Acesta permitea programarea rapidă a unor aplicații simple (în special jocuri) folosind instrucțiuni hexadecimal de nivel înalt. Deși sistemul original a devenit istoric, CHIP-8 rămâne un standard didactic util în dezvoltarea emulatorilor. Așa cum subliniază Langhoff, „sugestia comună este să pornești cu CHIP-8” atunci când înveți să construiești emulatoare, iar proiectele moderne evidențiază că CHIP-8 oferă un sistem “simplu, dar complet” pentru învățarea conceptelor de bază. Proiectul de față implementează un emulator CHIP-8 în C folosind SDL2, urmărind să exploreze arhitectura virtuală a acestui sistem (memorie, registre, instrucțiuni, I/O) şi să asigure rularea jocurilor originale CHIP-8. Obiectivul educațional este ca lucrarea să demonstreze înțelegerea principiilor fundamentale de arhitectură a calculatorului și programare grafică la nivel scăzut, integrate într-un proiect coerent.

**Scopul proiectului**

* Reproducerea arhitecturii CHIP-8: memorie de 4 KB (4096 octeți), 16 registre de 8 biți (V0–VF), registrul index I, contor de program PC și stivă de returnări.
* Implementarea completă a setului de instrucțiuni CHIP-8 (aproximativ 35 de instrucțiuni distincte), incluzând salturi, subrutine, operații aritmetice/bitwise și grafice.
* Afișarea grafică 2D folosind SDL2: ecran monocrom 64×32 pixeli conform specificațiilor CHIP-8, cu proceduri pentru desenarea sprite-urilor (DXYN) și curățarea ecranului (00E0).
* Gestionarea intrărilor de la tastatură: maparea tastelor standard pe cele 16 taste hex CHIP-8 și implementarea instrucțiunilor care testează starea tastelor (skip condiționat sau așteptare până la apăsare).
* Realizarea mecanismelor de temporizare: două contoare (delay timer și sound timer) care se decrementează la 60 Hz și declanșează efecte de sunet conform specificației.
* Obiectiv educațional: proiectul consolidează noțiuni de programare la nivel scăzut (manipulare memorie, lucrul cu biți și registre, structuri simple) și demonstrează abilități de integrare a librăriilor multimedia (SDL2) într-o aplicație funcțională.

**Spațiu de lucru**

A screen shot of a computer program

AI-generated content may be incorrect.Proiectul a fost dezvoltat integral în Linux (Void). Drept editor de text am folosit VIM, drept compilator CLANG (deoarece îl prefer peste GCC) și un fișier de tip script shell **build.sh** prin intermediul căruia compilez proiectul. Pentru version control am folosit github (<https://github.com/mday06/CHIP-8/tree/master>).

**Prezentarea arhitecturii CHIP-8**

* **A screenshot of a computer program

  AI-generated content may be incorrect.Memoria**: 4096 octeți de memorie adresabili. Zonele 0x000–0x1FF sunt rezervate (încărcător de sistem şi fonturi), iar programele CHIP-8 încep, convențional, de la 0x200. Sprite-urile font pentru cifrele 0–F sunt stocate în mod obișnuit între 0x50 și 0x9F.
* **Registre**: 16 registre de 8 biți V0–VF pentru operații generale. Registrul VF este folosit ca flag pentru operații aritmetice (transport, „no borrow”) sau coliziuni grafice. Există și un registru index I (16 biți) pentru adresare în memorie și contorul de program PC (16 biți) pentru următoarea instrucțiune.
* **Stivă**: Folosită pentru salvarea adreselor de retur la apelurile de subrutine. Implementările uzuale oferă cel puțin 16 niveluri de stivă (versiunea originală aloca 12 niveluri).
* **Display**: Ecran monocrom 64×32 pixeli. Fiecare pixel poate fi pe (alb) sau off (negru). Grafica se realizează desenând sprite-uri de lățime 8 pixeli și înălțime 1–15 pixeli; fiecare bit de 1 din sprite inversează (XOR) pixelul corespunzător pe ecran.
* **Tastatură (Keypad)**: 16 taste hex (0–F) aranjate pe o matrice 4×4. Instrucțiunile de tip EX9E/EXA1 sar instrucțiunea următoare condiționat de starea unei taste specifice, iar FX0A așteaptă o apăsare de tastă pentru a continua.
* **Timere**: Două contoare de 8 biți (delay timer și sound timer) care se decrementează cu 1 la fiecare 1/60 secunde. Când sound timer este nenul, sistemul emite un bip continuu. Instrucțiunile FX15/FX18 setează valoarea timpurilor, iar FX07 permite citirea acestora.

A screen shot of a computer program

AI-generated content may be incorrect.**Structura generală a aplicației**

* **Inițializare (SDL2)**: Se configurează biblioteca SDL2 pentru grafică și input: se creează o fereastră cu rezoluția logica 64×32 pixeli (adesea mărită pentru vizibilitate) și un *renderer*. Se resetează starea internă a emulatorului: toate registrele și memoria la 0, stiva goală și PC setat la 0x200. Se încarcă fonturile (sprite-urile pentru cifre) în memoria internă și se încarcă ROM-ul jocului ales la adresa 0x200.
* **Bucla principală (fetch–decode–execute)**: Emulatorul rulează într-o buclă continuă în care citește instrucțiunile CHIP-A computer screen shot of a program

  AI-generated content may be incorrect.8. La fiecare pas se preia (*fetch*) codul de 2 octeți de la adresa PC, se mărește PC cu 2, apoi se decodifică și se execută instrucțiunea. Decodificarea se realizează de obicei cu o structură switch-case sau un tabel de funcții, iar execuția modifică registre, memorie, stivă sau buffer-ul de ecran, după caz.

* A screenshot of a computer code

  AI-generated content may be incorrect.**Gestionarea ecranului**: Aplicația menține un buffer intern de 64×32 biți pentru display. Instrucțiunile grafice (DXYN, 00E0) modifică acest buffer. De fiecare dată când buffer-ul se schimbă (de exemplu după un DXYN sau după curățarea ecranului), conținutul său este randat în fereastra SDL, pixel cu pixel, folosind un textură sau operațiuni de desen 2D.
* **Intrare (input)**: Evenimentele de la tastatură sunt preluate de SDL2. Cele 16 taste CHIP-8 sunt mapate pe tastele fizice (de ex. 1..4, QWER, ASDF, ZXCV). La execuția instrucțiunilor EX9E/EXA1, emulatorul verifică starea tastei indicate şi sare condiționat peste următoarea instrucțiune. Pentru instrucțiunea FX0A, bucla principală rămâne într-o stare de așteptare până la apăsarea unei taste, moment în care stochează codul tastei în registrul Vx.
* **Timere și sunet**: Aplicația gestionează două contoare intermediare sincronizate cu aproximativ 60 Hz. Delay timer și sound timer se decrementează la fiecare 1/60 secunde. Dacă sound timer este >0, se generează un semnal sonor (de exemplu un *beep*) până la revenirea la 0. Emulatorul poate folosi SDL2 (sau SDL\_mixer) pentru redarea sunetului corespunzător.

**Explicarea funcționării**

* A screen shot of a computer program

  AI-generated content may be incorrect.**Inițializare**: La pornirea emulatorului, toate registrele V0–VF, precum și I și PC, sunt inițializate cu zero, iar PC este setat la 0x200 (adresa de start a programului). Bufferul de afișare este golit, iar cele două timere sunt setate la 0. Se încarcă sprite-urile fontului în memoria internă și se încarcă jocul (fișierul ROM) la adresa 0x200.
* **Ciclul de execuție**: Emulatorul intră într-o buclă continuă cu trei faze principale: *fetch*, *decode*, *execute*. Mai întâi, se citesc din memorie doi octeți de la poziția curentă PC şi se formează codul de 16 biți al instrucțiunii. PC este incrementat cu 2 pentru următoarea instrucțiune (în timp ce *execute* poate ajusta PC în plus pentru skip-uri sau sărituri). Instrucțiunea este decodificată prin analizarea primului nibble (de ex. 1NNN, 2NNN, 6XNN etc.), iar cazul corespunzător efectuează acțiunea: de exemplu, o instrucțiune de salt (1NNN) va seta PC la NNN, iar o instrucțiune de adunare (7XNN) va adăuga valoarea NN registrului Vx.
* A computer screen shot of a code

  AI-generated content may be incorrect.**Execuția instrucțiunilor grafice**: Instrucțiunea DXYN extrage N octeți (sprite) din memorie începând de la adresa stocată în I. Fiecare bit 1 din aceste octeți inversează starea pixelului corespunzător pe ecran (operație XOR). Dacă vreun pixel este înlăturat (de la 1 la 0), se consideră că a avut loc o coliziune și registrul VF primește valoarea 1; în caz contrar, VF rămâne 0. Instrucțiunea 00E0 (clear screen) resetează buffer-ul de ecran, setând toți pixeli la 0. După modificarea buffer-ului, acesta este redat în fereastra grafică pentru a reflecta schimbările.
* **Input utilizator**: Starea tastelor fizice este citită prin evenimente SDL2. La executarea lui EX9E, emulatorul verifică dacă tasta corespunzătoare registrului Vx este apăsată; dacă da, sare peste următoarea instrucțiune. Instrucțiunea EXA1 face saltul când tasta **nu** este apăsată. Instrucțiunea FX0A blochează execuția până când utilizatorul apasă o tastă; codul tastei apăsate este apoi stocat în registrul Vx. Astfel, jocul poate reacționa la input în mod sincron.
* **Timp și sunet**: La fiecare ciclu (sau la interval fix de timp), delay timer și sound timer sunt decrementate cu 1 dacă sunt nenule. Jocurile CHIP-8 pot sincroniza animații sau acțiuni prin valoarea delay timer. Când sound timer > 0, emulatorul emite un semnal sonor (beep) până când contorul ajunge la 0. Instrucțiunile FX15 și FX18 setează delay timer, respectiv sound timer, iar FX07 permite citirea valorii actuale a delay timer-ului.

**Concluzii**

Implementarea emulatorului CHIP-8 în C cu SDL2 a oferit o perspectivă practică asupra modului în care un calculator (virtual) simplu își gestionează resursele interne. Proiectul, de complexitate moderată (circa 35 instrucțiuni de tradus și integrarea unui subsistem grafic și de intrare), reflectă eficient combinarea conceptelor de arhitectură de calcul și programare în C la nivel scăzut. Prin dezvoltarea sa, mi-am antrenat capacitatea de a lucra cu manipulări de biți, cu structuri fundamentale (memorie, stivă) și de a utiliza biblioteca SDL2 pentru interfața grafică și tratarea tastelor.

**Bibliografie**

* Tobias V. Langhoff, *Guide to making a CHIP-8 emulator*, 2020 – ghid online detaliat (<https://tobiasvl.github.io/blog/write-a-chip-8-emulator/>).
* SDL – Simple DirectMedia Layer, site oficial (<https://www.libsdl.org>).
* Wikipedia, *CHIP-8* (articol enciclopedic) – descriere hardware/software (<https://en.wikipedia.org/wiki/CHIP-8>).