8mu

Specyfikacja implementacyjna

Krzysztof Piekarczyk
288277

26 marca 2020

Spis treści

1 Wstęp

Celem tego dokumentu jest opisanie implementacji funkcjonalności programu „8mu" opisanych w specyfikacji funkcjonalnej oraz opisu działań i narzędzi użytych w procesie implementacji.

2 Środowisko deweloperskie

2.1 Software

- System operacyjny: Windows 10
- Język programowania: C++11
- CLion 2019.3
- GCC-6.3.0-1

Projekt będzie wykorzystywał CMake.

2.2 Hardware

Program tworzony i testowany będzie na komputerze stacjonarnym o następującej specyfikacji:

- Procesor `Intel Core i7-4770K`
  - rdzenie: 4
  - wątki: 8
- 12 GB pamięci RAM
- Dyski twarde
  - `Samsung SSD RBX Series 128GB m` – 128GB
  - `Seagate Barracuda 7200.11 ST3320613AS` – 320GB
  - `Seagate ST1000DX001` – 1TB
  - `Western Digital Caviar Blue 500 WD5000AAKX` – 500GB
  - `Nieokreślonej produkcji dysk SSD` – 128GB

3 Zasady wersjonowania

3.1 Obsługa gałęzi

Nowe funkcjonalności wprowadzane będą na gałęzi `features` a następnie poprawnie działający kod jest wprowadzany do gałęzi `master`, co zapewnia, że funkcjonalności w gałęzi `master` zawsze są ukończone.

Jeżeli na gałęzi `master` znajdzie się błędnie działający kod, zostaje on poprawiony na gałęzi `fixes`.

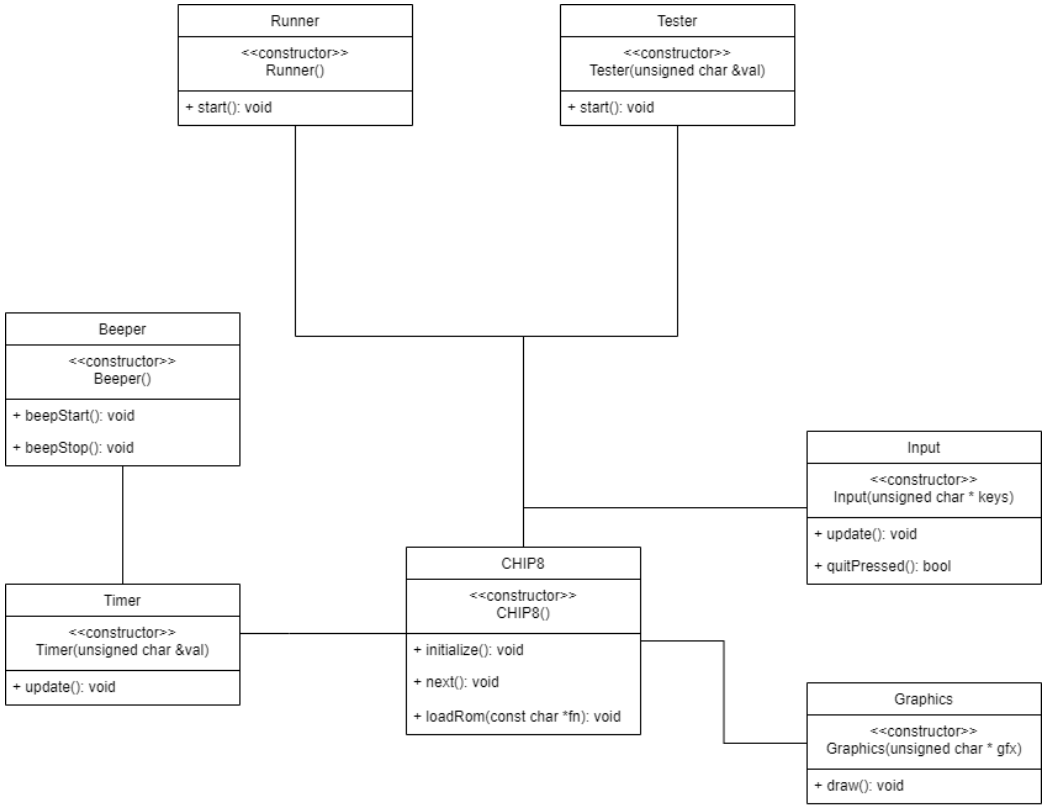
3.2 Szablon wiadomości

Wiadomości do repozytorium będą trzymać się danego szablonu:

- „`add *`" gdy dodany zostanie jakiś nowy plik/fragment kodu
- „`remove *`" gdy usunięty zostanie jakiś plik/fragment kodu
- „`modify *`" gdy zostanie zmieniony już istniejący fragment pliku/kodu
- „`fix *`" gdy zostanie naprawiony jakiś błąd w kodzie/formatowania plików
- „`qs`" w sytuacji w której potrzebne jest natychmiastowe odstąpienie od komputera, wiadomość tylko po to, żeby zapisać w repozytorium obecny stan

4 Klasy

4.1 Diagram klas



Rysunek 4.1: Diagram klas

4.2 Opis klas

- `CHIP8`: Klasa odpowiedzialna za wykonywanie wgranego kodu, implementuje maszynę wirtualną interpretera CHIP-8.
- `Beeper`: Klasa odpowiedzialna za wydanie jedynego dźwięku.
- `Graphics`: Klasa odpowiedzialna za wyświetlanie grafiki.
- `Input`: Klasa odpowiedzialna za wychwytywanie naciśnięć klawiszy.

- `Timer`: Klasa odpowiedzialna za obsługę zegarów, których w CHIP-8 są dwa (60Hz)
- `Runner`: Klasa odpowiedzialna za uruchomienie programu w trybie wykonywania zadanego kodu.
- `Tester`: Klasa odpowiedzialna za uruchomienie programu w trybie testowym.

5 Implementacja

Interpreter CHIP-8 opiera się o maszynę wirtualną o specyfikacji:

- dwubajtowy rejestr aktualnej instrukcji (`unsigned short opcode`)
- 4KiB pamięci (`unsigned char memory[4096]`)
  - programy wgrywane są do pamięci na pozycji 0x200, poniżej jest przestrzeń zarezerwowana dla interpretera, w tym wypadku zawierać będzie wbudowane czcionki
- 15 8b rejestrów ogólnego zastosowania (V0-VE) oraz jeden na flagę przeniesienia (VF)(`unsigned char V[16]`)
- licznik programu (wartości od 0x000 do 0xFFF więc mieści się w 2B) (`unsigned short pc`)
- rejestr indeksu (zakres taki jak w `pc`) (`unsigned short I`)
- 16sto poziomowy stos (`unsigned short stack[16]`)
- wskaźnik stosu (`unsigned char sp`)
- grafika
  - 64x32
  - monochromatyczna (piksel włączony lub wyłączony)
  - rysowanie odbywa się przy pomocy spritów
  - rysowanie odbywa się w trybie XOR, jeśli jakiś piksel zostaje wyłączony w trakcie rysowania ustawiana jest flaga VF, pozwala to na wykrywanie kolizji
- obsługa 16 klawiszy (0x0-0xF) (`unsigned char keys[16]`)
- dwa zegary
  - dekrementują swoją wartość o zakresie 0x00-0xFF z częstotliwością 60Hz
  - jeden z nich jest zegarem służacym do odliczania czasu
  - drugi wywołuje dźwięk, jeśli ma niezerową wartość

Po uruchomieniu programu odbywa się inicjalizacja rejestrów maszyny wirtualnej, zadany program wgrywany jest do pamięci maszyny od adresu 0x200. Po zakończeniu tego kroku, rozpoczyna się pętla emulacji:

1. odświeżenie stanu maszyny
2. pobranie instrukcji z pamięci
3. zdekodowanie instrukcji
4. wykonanie instrukcji

Język CHIP-8 posiada 35 kodów operacyjnych (instrukcji), których spis znajduje się m.in. pod adresem: https://github.com/mattmikolay/chip-8/wiki/CHIP-8-Instruction-Set.