

Spis treści

| | |
|--|----|
| Obsługa emulatora „Maszyny RAM” | 2 |
| 1.1. Informacje ogólne | 2 |
| 1.2. Składowe edytora Maszyny RAM | 3 |
| 2. Tabela statystyki..... | 4 |
| 3. Aktualny status programu | 5 |
| 4. Zawartość rejestrów | 7 |
| 5. Menu sterowania | 7 |
| 6. Taśma wejściowa..... | 13 |
| 7. Program RAM | 14 |
| 8. Taśma wyjściowa..... | 14 |
| 1.2 Weryfikacja i uruchomienie programu..... | 15 |
| 1.3 Obliczanie złożoności algorytmu | 18 |

Obsługa emulatora „Maszyny RAM”

1.1. Informacje ogólne

Rozdział został poświęcony przedstawieniu emulatora maszyny RAM stworzonej w ramach części praktycznej pracy magisterskiej. W rozdziale zostaną opisane najważniejsze elementy aplikacji potrzebne do zrozumienia działania emulatora. Opis będzie zawierał również instrukcje obsługi emulatora wraz z przykładami, które pozwolą na szybsze przyswojenie wiedzy na temat obsługi programu.

Aplikacja webowa, która emuluje działanie maszyny RAM służy do tworzenia, analizy (debugowania), testowania oraz uruchamiania programów napisanych w kodzie RAM. Stworzenie emulatora w formie aplikacji webowej niesie ze sobą jedną ważną cechę, a mianowicie proste przenoszenie aplikacji między różnymi systemami. Był to jeden z głównych czynników, który przyczynił się stworzenia emulatora w takiej formie. Wygląd emulatora może częściowo przypominać aplikacje okienkowe wykorzystywane do tworzenia oprogramowania na różne urządzenia i systemy. Miało to na celu zapewnienie prostej obsługi i szybkiej nauki obsługi programu.

Do działania aplikacji niewymagane są duże zasoby sprzętowe. Do uruchomienia aplikacji wymagana jest jedynie przeglądarka internetowa, na której stworzony emulator będzie obsługiwany. Strona może zostać umieszczona również na serwerze bez potrzeby przenoszenia jej przy pomocy zewnętrznych nośników. Samo działanie emulatora zostało przetestowane na przeglądarkach takich jak Google Chrome (wersja 92.0.4515.107), Microsoft Edge (wersja 92.0.902.55) oraz Firefox (90.0.2). Aplikacja bez większych problemów powinna zostać uruchomiona również na starszych i nowszych wersjach przeglądarek.

Należy pamiętać o tym, że abstrakcyjny model maszyny RAM nie posiada ograniczeń wynikających z liczby komórek pamięci, komórek taśmy wejściowej i wyjściowej oraz maksymalnego rozmiaru danych, jakie można wpisać do pojedynczej komórki. Sama aplikacja została napisana w taki sposób, aby również nie nakładała ograniczeń na dane wprowadzane przez użytkownika (oprócz liczby iteracji programu, po których emulator określa, czy program wszedł w nieskończoną pętlę). Program jednak zostanie ograniczony przez zasoby urządzenia oraz system operacyjny, na którym zostanie uruchomiona aplikacja.

Oprócz podstawowych funkcjonalności jakie udostępnia emulator tj. tworzenie własnych programów i ich uruchamianie, program pozwala również na analizowanie złożoności obliczeniowej, dzięki zawartym funkcjom. Emulator pozwala na obliczanie złożoności algorytmów w czasie rzeczywistym (wartość złożoności jest aktualizowana podczas

wykonywania każdej instrukcji w kodzie RAM). Dodatkowo użytkownik ma możliwość określenia złożoności dla różnych wartości taśmy wejściowej i przedstawienie otrzymanych wyników w formie wykresu. Zastosowane funkcje wspomogą analizę algorytmów i pozwolą na dobranie najbardziej optymalnego rozwiązania w zależności od potrzeb.

Językiem programowania, na którym opiera się główna logika działania emulatora jest język JavaScript. Jest to jeden z głównych i najbardziej popularnych języków do tworzenia stron i aplikacji webowych. Sam wygląd strony został stworzony w języku HTML i CSS. Poprzez zastosowanie prostych rozwiązań tworzona aplikacja powinna być kompatybilna z większościami przeglądarek. Dodatkowo do tworzenia wykresów przedstawiających złożoność algorytmu dla różnych danych wejściowych wykorzystano bibliotekę stworzoną przez Autora strony chart.js. Biblioteka została stworzona w języku JavaScript i udostępnia dużą ilość gotowych funkcji i stylów do tworzenia własnych spersonalizowanych wykresów. Wygląd strony został przedstawiony na rysunku 6.1.

1.2. Składowe edytora Maszyny RAM

W skład emulatora maszyny RAM wchodzi elementy przedstawione na rysunku 6.1. W skład emulatora wchodzi: (1) Tabela statystyki, (2) Aktualny status programu, (3) Zawartość rejestrów maszyny RAM, (4) Menu sterowania, (5) Taśma wejściowa, (6) Program RAM, (7) Taśma wyjściowa. Każdy z elementów zostanie opisany dokładnie w poniższych rozdziałach.

The screenshot shows the 'Emulator maszyny RAM' interface. At the top is a green header with the logo of Politechnika Rzeszowska, the title 'Emulator maszyny RAM', and the author 'Łukasz Janik'. Below the header is a toolbar with icons for file operations and execution. The main interface is divided into several sections:

- Statystyki:** A table showing statistics like 'Licznik rozkazów' (0), 'Licznik wykonanych rozkazów', 'Zużycie czasu', 'Zużycie czasu (Log)', and 'Zużycie pamięci'. This is marked with a red circle 1.
- Status:** A section for the current program status, marked with a red circle 2.
- Zawartość rejestrów:** A section for the contents of the machine registers, marked with a red circle 3.
- Menu sterowania:** A toolbar with icons for file operations and execution, marked with a red circle 4.
- Taśma wejściowa:** A section for the input tape, showing 'Długość taśmy wejściowej: 0' and a row of input boxes numbered 1 to 6. This is marked with a red circle 5.
- Program RAM:** A table with columns: Nr, Etykieta, Komenda, Adres, Komentarz, and Operacje. The 'Komenda' column has a dropdown menu set to 'LOAD'. This is marked with a red circle 6.
- Taśma wyjściowa:** A section for the output tape, showing 'Długość taśmy wyjściowej: 0' and a row of output boxes numbered 1 to 6. This is marked with a red circle 7.

Rys. 0.1 Wygląd emulatora maszyny RAM

2. Tabela statystyki

Tabela statystyki (oznaczona numerem 1) służy do wyświetlania informacji o złożoności algorytmów oraz podstawowych informacji o ilości wykonanych rozkazów, a także liczbie poszczególnych instrukcji kodu RAM wykonanych podczas działania programu. Dane są wyświetlane dopiero po uruchomieniu kodu przez użytkownika i tyczą się liczb wprowadzonych na taśmie wejściowej. Należy w tym miejscu zauważyć, że program może być wykonywany w sposób ciągły, tzn. program wykonywany do momentu wykrycia przez emulator komendy HALT, która kończy działanie programu RAM. Drugim sposobem na rozpoczęcie pracy programu RAM jest wykonywanie kolejnych instrukcji krok po kroku. Oznacza to, że program wykona jedną instrukcję i się zatrzyma do momentu aż użytkownik nie wymusi wykonania kolejnej instrukcji. Ostatnią metodą, która może być przydatna między innymi do wykrywania błędów programu jest wykonanie programu z użyciem „breakpointów”. W takiej sytuacji program wykonuje kolejne instrukcje do momentu aż wykryje komendę oznaczoną „breakpointem” i zostaje w niej do momentu wykonania akcji przez użytkownika (wznowienia działania programu). W przypadku analizy tabeli „Statystyki” należy zwrócić uwagę na dwie ostatnie metody. Jeżeli program nie zostanie wykonany do końca, tzn. zostanie zatrzymany na jednej z instrukcji, to dane w tabeli są wyświetlane dla wykonanego fragmentu kodu, a nie całości (czyli do wykonania komendy HALT). Należy zwrócić szczególną uwagę na takie sytuacje, ponieważ może doprowadzić to do licznych błędów podczas analizy.

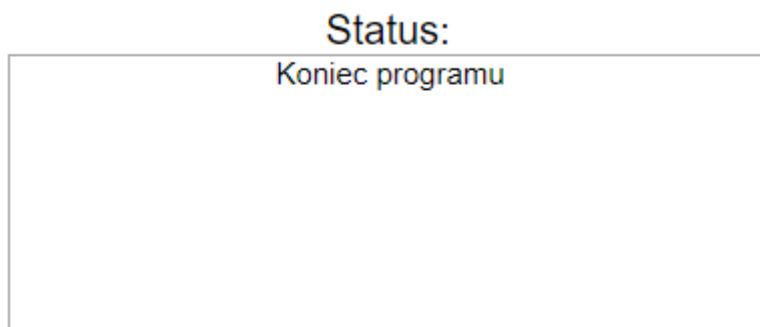
Statystyki:

| | |
|-----------------------------|---------------------|
| Licznik rozkazów | 15 |
| Licznik wykonanych rozkazów | 30 |
| Zużycie czasu | 31 |
| Zużycie czasu (Log) | 80 |
| Zużycie pamięci | 3 |
| Zużycie pamięci(Log) | 7 |
| | LOAD: 8 STORE: 1 |

Rys. 0.2 Tabela statystyki

3. Aktualny status programu

Kolejnym ważnym elementem emulatora maszyny RAM jest okienko „Status”. Okno służy do wyświetlania wszystkich informacji o aktualnym statusie, błędach oraz o zakończeniu wykonywania programu. Podczas próby uruchomienia błędnego programu wprowadzonego przez użytkownika symulacja zostanie zatrzymana. Dodatkowo zostanie wyświetlony komunikat, który naprowadzi użytkownika na błąd w kodzie. Na rysunku 6.3 przedstawiono komunikat wyświetlany w przypadku prawidłowego wykonania całego programu.



Rys. 0.3 Wygląd okienka Status

Poniżej została umieszczona lista komunikatów wyświetlana w okienku „Status” podczas analizy i wykonywania programu w kodzie RAM:

- „Koniec programu” – wykonywanie programu zakończyło się pomyślnie
- „Przekroczono czas działania programu” – w przypadku gdy liczba iteracji programu przekroczy 100 000 rozkazów i nie zakończył swego działania – program wpadł w nieskończoną pętlę
- „Program wstrzymano na instrukcji nr: x” – komunikat wyświetlany w przypadku wykonywania programu przy użyciu „breakpointów”. W miejscu x wpisywany jest numer instrukcji, na której program został zatrzymany
- „Brak etykiety x” – informacja wyświetlana w przypadku, gdy emulator nie znajdzie etykiety, do której powinien skoczyć (podanej jako adres komendy skoku)
- „Taśma wejścia się skończyła” – komunikat wyświetlany dla przypadku, gdy program wykonuje operacje READ, lecz na taśmie wejściowej nie napotkał żadnej liczby
- „Brak komendy HALT” – komunikat wyświetlany w przypadku braku komendy HALT w programie

- „Wykryto błąd w instrukcji x” – zawartość wyświetlana w przypadku wykrycia niedozwolonej operacji w kodzie RAM np. wpisanie adresu do komendy HALT
- „Błędny numer instrukcji” – sprawdzanie numeru instrukcji – w przypadku wczytywania danych z pliku .txt lub .csv
- „Dla komendy x nie można użyć operandu y” – komunikat wyświetlany w sytuacji, gdy zostanie wykonana niedozwolona akcja dla instrukcji niezgodna z kodem RAM
- „Wykryto błędną instrukcję” – komunikat wyświetlany dla sytuacji, gdy komenda podana przez użytkownika nie jest zgodna z podstawowymi komendami w standardzie języka RAM – komunikat przydatny głównie podczas wczytywania plików z instrukcjami
- „Etykieta x nie może rozpoczynać się od cyfry” – według standardu kodu RAM etykiety nigdy nie mogą zaczynać się od cyfry
- „Etykieta x została już użyta” – nie można użyć dwa razy tej samej nazwy etykiety. Etykiety powinny posiadać niepowtarzalną nazwę
- „Adres rejestru musi być liczbą dodatnią” – komórki w rejestrach zawsze muszą mieć wartość dodatnią
- „Błędny operand x” – komunikat wyświetlany w przypadku użycia innego operandu niż zdefiniowane w kodzie RAM. W emulatorze rozróżniamy 3 typy operandów: *, = i, " "(pusty), które zostały opisane w rozdziale 3
- „Adres x musi być etykietą w przypadku wybrania instrukcji JUMP, JZERO lub JGTZ” – komunikat informuje o potrzebie użycia etykiety w przypadku instrukcji skoku
- „Dla instrukcji y nie można użyć etykiety” - w polu adres można użyć etykiety jedynie dla instrukcji skoku
- „Instrukcja HALT nie może zawierać adresu” – komunikat informujący o braku możliwości użycia adresu dla komendy HALT
- „Nie można poprawnie odczytać adresu instrukcji nr x” – w przypadku błędu podczas sprawdzania adresu (inny błąd niż uwzględniony w komunikatach powyżej)
- „Błędna instrukcja” – dodatkowy komunikat informujący o błędzie (zazwyczaj wyświetlany jako dodatkowy komunikat)

4. Zawartość rejestrów

Tabela z zawartością rejestrów wyświetla aktualną zawartość komórek w rejestrze. Kolumna pierwsza przedstawia indeksy komórek pamięci, do których może odwoływać się użytkownik. Należy pamiętać, że indeksy komórek pamięci mogą przyjmować tylko wartość dodatnią. Komórka o indeksie 0 zwana jest akumulatorem i jest wykorzystywana przy większości operacji obliczeń. Ilość indeksów wyświetlanych dla użytkownika może być różna, ze względu na zastosowany program. Jeżeli liczba komórek pamięci jest duża tabela zamienia się w listę przesuwaną, która ułatwia przeglądanie zawartości. Komórki pamięci, które nie są wyświetlane posiadają wartość nieokreśloną dlatego też nie zostają wyświetlone.

| Zawartość rejestrów | |
|---------------------|---|
| Rejestr 0: | 0 |
| Rejestr 1: | 0 |
| Rejestr 2: | 5 |

Rys. 0.4 Wygląd tabeli z zawartością komórek rejestru

5. Menu sterowania

Kolejnym ważnym elementem programu są przyciski przeznaczone do kontroli emulatora. Przyciski służą m.in. do wczytania lub zapisania programu w kodzie RAM, uruchamiania programu, debugowania oraz analizy złożoności algorytmu. Przyciski posiadające kolor zielony są aktywne i można je użyć w dowolnym momencie działania programu. W przypadku szarej ramki przyciski są nieaktywne i próba użycia ich nie spowoduje żadnej akcji. Sytuacja nieaktywnych przycisków dotyczy głównie elementów służących do uruchamiania programu w kodzie RAM i występuje, gdy program zakończy swoje działanie. Aby ponownie aktywować przyciski, należy zresetować program. W dalszej części rozdziału zostanie omówiona każda dostępna funkcjonalność menu sterowania.



Rys. 0.5 Wygląd menu sterowania

Pierwszy z omawianych przycisków przedstawiony na rysunku 6.6 służy do wczytania nowego projektu. Powoduje on wyczyszczenie całej zawartości programu wraz z taśmą wejściową, rejestrami oraz tabelą statystyki. Przed jego użyciem należy upewnić się, że program, który stworzyliśmy podczas danej sesji został zapisany. W przeciwnym razie program zostanie

utracony. Jako dodatkowe zabezpieczenie przed przypadkowym skasowaniem danych sesji jest wyskakujący komunikat, który sprawdza, czy operacja wykonywana przez użytkownika jest świadomym działaniem. Po zatwierdzeniu komunikatu można przystąpić do tworzenia nowego programu w kodzie RAM.



Rys. 0.6 Przycisk "Nowa karta"

Kolejny przycisk patrząc od lewej strony (przedstawiony na rysunku 6.7) służy do zapisu programu napisanego w kodzie RAM do pliku z rozszerzeniem .txt lub .csv. Po jego naciśnięciu zostanie wyświetlone okienko pokazane na rysunku 6.8. Okno zawiera podgląd kodu, który zostanie zapisany w pliku. Format zapisu został określony przez autora pracy w taki sposób, aby był czytelny i łatwy do analizy. Aby program mógł zostać ponownie wczytany do emulatora, należy pozostawić plik w niezmienionej formie. Poniżej podglądu pliku znajdują się pole do wpisania nazwy pliku oraz wyboru formatu, w jakim plik powinien zostać zapisany. W przypadku braku nazwy pliku zostanie wyświetlony komunikat o błędzie. Jeżeli wymagane pola zostaną wypełnione można przejść do generowania pliku. W tym celu należy nacisnąć przycisk „Wygeneruj plik”. Akcja ta spowoduje wyświetlenie linku poniżej przycisku, za pomocą którego można pobrać plik.



Rys. 0.7 Przycisk "Zapisz zawartość"


```

nr,etykieta,komenda,operand,adres,komentarz;
1,,READ,,1,;
2,,LOAD,,1,;
3,,STORE,,2,;
4,DOPOKI,LOAD,,1,;
5,,JZERO,,koniec_dopoki,;
6,,LOAD,,1,;
7,,SUB,,2,;
8,,JGTZ,,nowy_max,;
9,,JUMP,,czytaj_x,;
10,nowy_max,LOAD,,1,;
11,,STORE,,2,;
12,czytaj_x,READ,,1,;
13,,JUMP,,DOPOKI,;
14,koniec_dopoki,WRITE,,2,;
15,,HALT,,,;

```

Nazwa pliku:

Format:

Wygeneruj plik

Rys. 6.8 Okienko do zapisu zawartości programu RAM

Przycisk przedstawiony na rysunku 6.9 służy do odczytu zawartości programu RAM z pliku z rozszerzeniem .txt lub .csv. Tak jak wspomniano wcześniej plik musi mieć dokładnie określoną strukturę. Każda instrukcja powinna zaczynać się od nowej linii, natomiast elementy wchodzące w skład instrukcji np. adres, operand czy etykieta muszą być rozdzielone przecinkami. Kolejność elementów nie jest przypadkowa, dlatego też należy przyjąć kolejność: (1) numer instrukcji, (2) operand, (3) komenda, (4) operand, (5) adres, (6) komentarz. Ważnym elementem jest również średnik na końcu komentarza, który pomaga w określeniu końca instrukcji oraz długości komentarza.



Rys. 0.9 Przycisk do wczytania programu

Na rysunku 6.10 przedstawiono wygląd okna do wczytywania plików z kodem RAM. Do wyboru są dwie opcje. Pierwsza z nich pozwala na wczytanie programu prosto z urządzenia, na którym jest uruchomiony program oraz druga pozwalająca wybrać jeden z trzech przykładowych programów, które są ukryte w kodzie emulatora. W przypadku wczytywania pliku z komputera należy kliknąć przycisk „Wybierz plik”, który wywoła okno systemowe, za

pomocą którego wskazujemy plik, który chcemy wczytać. Gdy plik jest wczytany możemy nacisnąć przycisk „Załaduj kod”, który spowoduje sprawdzenie poprawności instrukcji zawartych w kodzie, a także załaduje kod do tabeli nr 6 pokazanej na rysunku 6.1 (jeżeli wszystkie instrukcje są poprawne). W przypadku przykładów, które zostały dostarczone razem z programem można znaleźć:

- „Oblicz n^n ” – Program pobiera z pierwszej komórki taśmy wejściowej liczbę, a następnie wykonuje operacje potęgowania np. wprowadzenie liczby 5 spowoduje wykonanie działania 5^5
- „Oblicz silnie” – Program pobiera liczbę z pierwszej komórki taśmy wejściowej i wykonuje operacje obliczania silni np. $5!$
- „Sumowanie liczb” – Program pobiera kolejne liczby z taśmy wejściowej i je sumuje do momentu aż na taśmie wejściowej pojawi się 0

Wybierz plik do odczytu (rozszerzenie txt lub csv):

Wybierz plik a następnie naciśnij przycisk "Załaduj kod"

lub

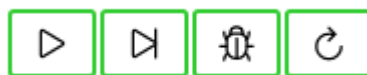
Załaduj gotowy przykład

Rys. 0.10 Okno wczytaj plik

Kolejna grupa przycisków pokazana na rysunku 6.11 służy do uruchamiania programu napisanego w kodzie RAM. Sposób ich użycia zostanie dokładnie przedstawiony w podrozdziale 6.3 *Weryfikacja i uruchomienie programu*. Poniżej został przedstawiony ogólny opis przycisków.

Pierwszy przycisk (zaczynając od lewej strony) służy do uruchomienia programu w kodzie RAM do momentu aż zostanie wykryta instrukcja HALT. Jeżeli program jest poprawny praca programu w kodzie RAM zawsze kończy się na instrukcji HALT. Kolejny przycisk pozwala na wykonywanie kolejnych instrukcji krok po kroku. Oznacza to, że wykonanie każdej kolejnej instrukcji programu wymaga wykonania akcji przez użytkownika (ponowne wciśnięcie przycisku). Trzeci przycisk służy do tzw. debugowania programu. W tym trybie

program zostanie zatrzymany w przypadku wykrycia „breakpoint” przy jednej z instrukcji. Dalsze wykonywanie instrukcji zostanie wznowione po ponownym naciśnięciu przycisku. Należy zauważyć, że w przypadku braku „breakpointów” przy instrukcjach program wykona się w taki sam sposób jak dla trybu pierwszego. Ostatni przycisk odpowiada za zresetowania stanu programu. Po użyciu tego przycisku wszystkie zmiany wprowadzone przez program np. zawartość rejestrów zostanie usunięta.



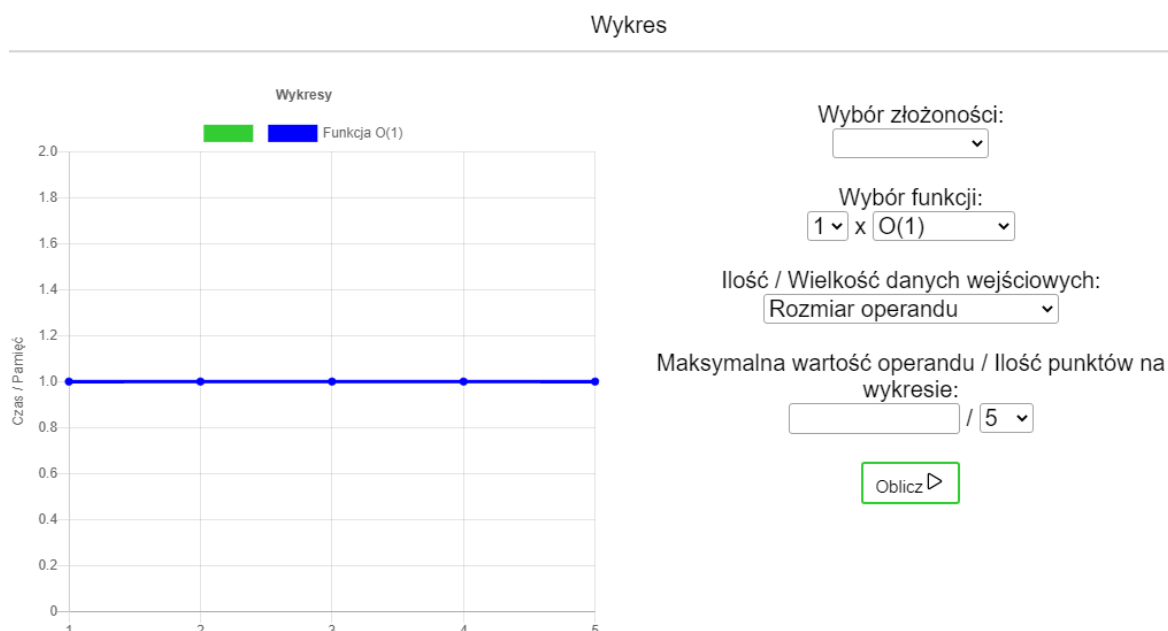
Rys. 0.11 Przyciski rozpoczynające pracę programu napisanego w kodzie RAM

Ważnym elementem emulatora jest przycisk przedstawiony na rysunku 6.12. Przyciski ten wyświetla okienko, za pomocą którego można generować wykresy przedstawiające poszczególne złożoności algorytmów dla zużytego czasu i pamięci zarówno dla zrównoważonego kryterium wagowego, jak i kryterium logarytmicznego. Wykresy mogą zostać porównane z wykresami funkcji np. $f(x)$ pozwalając na określenie wykresu, który najlepiej opisuje daną złożoność.



Rys. 0.12 Przycisk do generowania wykresów złożoności algorytmów

Na rysunku 6.13 został przedstawiony przykładowy wygląd okna wykresów. Po prawej stronie znajduje się pole wykresu przedstawiające wyniki obliczeń w postaci wykresu liniowego. Kolorem zielonym zawsze zostaje oznaczony wykres złożoności wybrany przez użytkownika przy pomocy listy wyboru „Wybór złożoności”. Kolorem niebieskim natomiast oznacza się wykres funkcji wybieranej w liście „Wybór funkcji”. Kolejna lista pozwala na dokonanie decyzji, co będzie rozmiarem zadania. Rozmiar zadania określa na jaką cechę taśmy wejściowej należy zwrócić uwagę. Rozmiarem zadania może być np. wielkość taśmy wejściowej lub rozmiar operandu. To użytkownik na podstawie wprowadzonego programu decyduje o wyborze. Określenie rozmiaru zadania jest bardzo ważne podczas generowania funkcji. Poniżej okna wyboru rozmiaru zadania znajdują się okienka zmieniane w zależności od wybranego typu badanego wejścia. Szczegółowy opis poszczególnych funkcjonalności dla wykresów złożoności znajduje się w rozdziale 6.4. Na samym końcu znajduje się przycisk do rozpoczęcia obliczania złożoności na podstawie wprowadzonych danych.



Rys. 0.13 Wygląd okna do generowania wykresów złożoności algorytmów

Ostatnią grupą przycisków przedstawiona na rysunku 6.13 pełni funkcję informacyjną. Pierwszy z nich w osobnej karcie wyświetla instrukcję obsługi emulatora maszyny RAM. Drugi przycisk natomiast wyświetla okienko z informacjami o autorze pracy itp. Okno to zostało przedstawione na rysunku 6.14. Oba przyciski nie wpływają na funkcjonowanie maszyny RAM.



Rys. 0.14 Przyciski do wyświetlania informacji oraz instrukcji

Informacje

Temat pracy

Emulator maszyny Abstrakcyjnej

Autor pracy

Łukasz Janik

Promotor

dr inż. Dariusz Rzońca

Rok akademicki

2021

Rys. 0.15 Okienko informacyjne

6. Taśma wejściowa

Kolejnym ważnym elementem emulatora maszyny RAM jest taśma wejściowa przedstawiona na rysunku 6.16. Taśma pozwala na wprowadzanie wartości do programu napisanego w kodzie RAM. Po wpisaniu wartości do pierwszej z komórek taśmy zostanie odblokowana możliwość wprowadzenia kolejnej wartości (początkowo okienka do wprowadzania danych są nieaktywne ze względu na zachowanie kolejności wprowadzania danych – nie można zostawić niezdefiniowanych pól w taśmie wejściowej). Ze względu na ograniczoną szerokość elementów taśmy, którą można wyświetlać jednocześnie zastosowano dwa przyciski do przesuwania taśmy w lewo lub prawo. Nad taśmą wyświetlana jest aktualna liczba elementów wprowadzona do poszczególnych komórek. Informacja o ilości wypełnionych komórek przydatna jest w przypadku wprowadzenia większej liczby elementów na taśmę wejściową.

Taśma wejściowa
Długość taśmy wejściowej: 0

| | | | | | | | |
|---|----------------------|----------------------|----------------------|----------------------|----------------------|----------------------|---|
| < | 1 | 2 | 3 | 4 | 5 | 6 | > |
| | <input type="text"/> | <input type="text"/> | <input type="text"/> | <input type="text"/> | <input type="text"/> | <input type="text"/> | |

Rys. 6.16 Taśma wejściowa

7. Program RAM

W celu wprowadzania instrukcji kodu RAM do emulatora została stworzona specjalna tabela przedstawiona na rysunku 6.17. Za pomocą tabeli można dodawać, edytować lub usuwać instrukcje w dowolnym momencie. Pierwsza kolumna używana jest podczas debugowania programu w kodzie RAM. Po dodaniu instrukcji przez użytkownika można zaznaczyć to okienko w celu zatrzymania programu przy danej instrukcji np. w celu znalezienia błędu. Druga kolumna określa numer instrukcji. Kolejne kolumny są powiązane bezpośrednio z programem w kodzie RAM i zostaną omówione w rozdziale 6.3. Ostatnia kolumna służy do dodawania i modyfikacji instrukcji.

|  | Nr | Etykieta | Komenda | | Adres | Komentarz | Operacje |
|---|----|----------------------|---------|---|----------------------|----------------------|----------|
| | | <input type="text"/> | LOAD ▾ | ▾ | <input type="text"/> | <input type="text"/> | + |

Rys. 0.17 Tabela do wprowadzania instrukcji w kodzie RAM

8. Taśma wyjściowa

Ostatnim ważnym elementem emulatora maszyny RAM jest taśma wyjściowa przedstawiona na rysunku 6.18. Taśma służy do wyświetlania danych wyjściowych programu stworzonego w kodzie RAM. Wartości są wypisywane przez algorytm za pomocą instrukcji WRITE do kolejnych komórek taśmy wyjściowej. Ze względu na ograniczoną szerokość elementów taśmy, którą można wyświetlać jednocześnie, zastosowano dwa przyciski do przesuwania taśmy w lewo lub prawo. Nad taśmą wyświetlana jest aktualna liczba elementów wypisanych do poszczególnych komórek taśmy. Informacja o ilości zapełnionych komórek przydatna jest w przypadku wprowadzenia większej liczby elementów na taśmę wyjściową.

Taśma wyjściowa
Długość taśmy wyjściowej: 0

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | |
| < | | | | | | | > |

Rys. 0.18 Taśma wyjściowa

1.2 Weryfikacja i uruchomienie programu

W tym rozdziale zostały opisane najważniejsze elementy emulatora wykorzystywane do tworzenia własnego kodu. Uwaga zostanie skupiona głównie na dodawaniu, modyfikacji oraz usuwaniu instrukcji. Pokazane również zostanie jak uruchamiać program w kodzie RAM.

Aby dodać instrukcję do tabeli z instrukcjami, należy użyć przycisku przedstawionego na rysunku 6.19. Domyślnie wybrana jest instrukcja LOAD bez żadnych parametrów. Przed użyciem przycisku należy uzupełnić pola, które są potrzebne i użyć wspomnianego przycisku.



Rys. 2.1 Przycisk do dodawania nowej instrukcji

W przypadku gdy dodaliśmy instrukcję, kolumna operacje zmienia się dla danego wiersza w sposób przedstawiony na rysunku 6.20. Pierwszy z przycisków służy do dodania nowej instrukcji poniżej instrukcji, dla której użyto przycisku „+”. Przycisk jest przydatny, gdy użytkownik zapomni wprowadzić jednej z instrukcji do kodu. W takim wypadku nie trzeba usuwać wszystkich komend z tabeli do miejsca, w którym chcemy dodać nową instrukcję. Wystarczy, że użytkownik doda nowy element pomiędzy już istniejące instrukcje. Kolejny przycisk służy do trwałego usuwania zbędnej instrukcji z tabeli. Należy uważać na użycie tego przycisku ze względu na to, że emulator nie pozwala przywrócić ostatnich zmian w kodzie. Ostatni przycisk używany jest do modyfikacji już dodanej instrukcji. Po jego naciśnięciu użytkownik może zmienić wszystkie elementy w wierszu.



Rys. 6.2 Kolumna „Operacje” dla dodanej instrukcji

Na rysunku 6.21 pokazano jak wygląda wiersz z instrukcją, który użytkownik chce zmodyfikować. Na tym etapie użytkownik może wprowadzić własne zmiany w instrukcji i zatwierdzić przyciskiem „OK”. W przypadku, gdy chcemy porzucić ostatnie modyfikacje należy kliknąć przycisk „Anuluj”.



Rys. 2.3 Wygląd wiersza instrukcji kodu RAM podczas modyfikacji

W przypadku każdego programu stworzonego w języku RAM w celu wprowadzenia wartości do programu używa się taśmy wejściowej oraz instrukcji READ. W przypadku, gdy nie wprowadzimy żadnych danych wejściowych do taśmy, to program zostanie zatrzymany na instrukcji READ. Wyświetlony zostanie również komunikat w okienku „Status” o braku wartości na taśmie wejściowej. Należy pamiętać, aby uzupełnić komórki taśmy wejściowej przed uruchomieniem programu, tak jak jest to pokazane na rysunku 6.22. Również należy zwrócić uwagę na to, że część komórek tablicy wejściowej jest zablokowana. Wynika to z faktu, że wartości na taśmie powinny zostać wprowadzane jedna po drugiej (nie można zostawić niezdefiniowanej komórki pomiędzy dwiema wartościami). Przyczyną takiego zachowania jest sama logika działania maszyny RAM. Głowica wczytuje kolejne liczby zgodnie z indeksami dlatego należy zwrócić na to szczególną uwagę.

Taśma wejściowa
Długość taśmy wejściowej: 4

| | 1 | 2 | 3 | 4 | 5 | 6 | |
|---|---|---|---|---|---|---|---|
| < | 1 | 2 | 5 | 0 | | | > |

Rys. 2.4 Wygląd uzupełnionej taśmy wejściowej


Po utworzeniu programu i wprowadzeniu danych do komórek taśmy wejściowej można przystąpić do pierwszego uruchomienia programu. Do celów demonstracyjnych w omawianym przykładzie został użyty gotowy program dostarczony razem z emulatorem służący do sumowania liczb z taśmy wejściowej. Aby szybko przetestować utworzony program można użyć przycisku , który uruchomi funkcje sprawdzające kod i w przypadku błędu wyświetli komunikat w okienku „Status”. W przypadku krótkiego programu, który jest prosty w analizie może użyć przycisku  którego naciśnięcie powoduje przejście do kolejnej instrukcji

programu. Aktualnie wykonywany wiersz instrukcji w tabeli zostanie oznaczony kolorem zielonym jak pokazano na rysunku 6.23. Przeładowana zostanie również wartość rejestrów.

| | | | | | | |
|----------------------------------|---|--------|-------|--------|-------------------------------------|---|
| <input checked="" type="radio"/> | 1 | CZYTAJ | READ | 1 | wczytaj kolejną wartość z taśmy | <input type="text" value="+"/> <input type="text" value="-"/> <input type="button" value="Edytuj"/> |
| <input type="radio"/> | 2 | | LOAD | 1 | | <input type="text" value="+"/> <input type="text" value="-"/> <input type="button" value="Edytuj"/> |
| <input type="radio"/> | 3 | | JZERO | KONIEC | jeśli wartość = 0 zakończ sumowanie | <input type="text" value="+"/> <input type="text" value="-"/> <input type="button" value="Edytuj"/> |


Rys. 2.5 Oznaczanie aktualnie wykonywanej instrukcji

Ostatnim sposobem na analizowanie kodu jest wstawianie „breakpointów” do instrukcji, w których chcemy znać wartość rejestrów oraz instrukcji, które będą wykonywane w kolejnych krokach. Analiza przy pomocy „breakpointów” może być bardziej przydatna w przypadku większych programów lub większej ilości danych wejściowych, gdzie wykonywanie instrukcji krok po kroku jest bardziej czasochłonne.

Na rysunku 6.24 przedstawiono instrukcje numer 3, dla której wstawiono „breakpoint”. Zaznaczenie odbywa się poprzez jedno kliknięcie na pierwszą kolumnę danej instrukcji. Usunięcie „breakpointów” również odbywa się poprzez jednokrotne kliknięcie. W celu uruchomienia tego trybu należy użyć przycisku . Po zatrzymaniu programu użytkownik musi ponownie użyć tego przycisku w celu dalszego uruchomienia programu. Emulator nie posiada ograniczeń wynikających z ilości występujących „breakpointów”

|  | Nr | Etykieta | Komenda | Adres | Komentarz | Operacje |
|---|----|----------|---------|--------|-------------------------------------|---|
| <input checked="" type="radio"/> | 1 | CZYTAJ | READ | 1 | wczytaj kolejną wartość z taśmy | <input type="text" value="+"/> <input type="text" value="-"/> <input type="button" value="Edytuj"/> |
| <input type="radio"/> | 2 | | LOAD | 1 | | <input type="text" value="+"/> <input type="text" value="-"/> <input type="button" value="Edytuj"/> |
| <input checked="" type="radio"/> | 3 | | JZERO | KONIEC | jeśli wartość = 0 zakończ sumowanie | <input type="text" value="+"/> <input type="text" value="-"/> <input type="button" value="Edytuj"/> |
| <input type="radio"/> | 4 | | ADD | 2 | dodaj dotychczasową | <input type="text" value="+"/> <input type="text" value="-"/> <input type="button" value="Edytuj"/> |

Rys. 2.6 Sposób zaznaczania "breakpointów"


Po uruchomieniu jednego z trybów nie ma możliwości przejścia do innego trybu pracy (reszta przycisków jest blokowana). Aby odblokować możliwość wyboru należy zrestartować program używając przycisku .

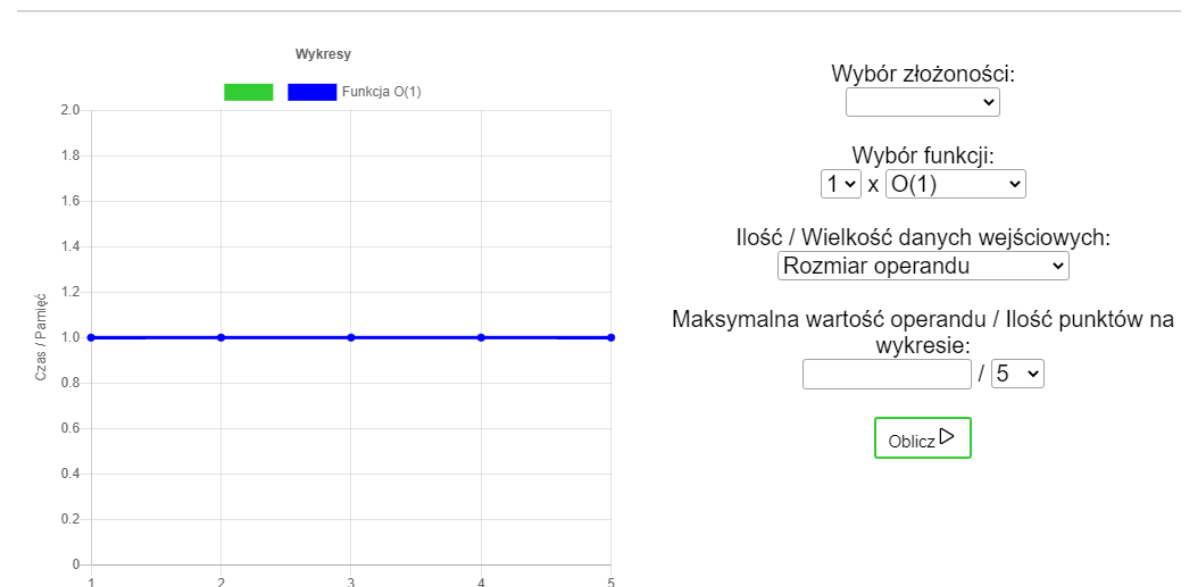
1.3 Obliczanie złożoności algorytmu

W celu analizy złożoności algorytmów utworzono dwa elementy w opisywanym emulatorze. Pierwszy z nich to tabela „Statystyki”, która pozwala na odczytanie podstawowych wartości analizy złożoności. Drugim elementem jest wykres generowany dla różnych wartości taśmy wejściowej, co pozwala na dokładniejszą analizę algorytmu oraz znalezienie funkcji opisującej dany program.

W przypadku tabeli „Statystyki” nie jest wymagana dłuższa analiza. Wartości wchodzące w skład tabeli to :

- „Licznik rozkazów” – liczba zapisanych rozkazów w kodzie RAM
- „Licznik wykonanych rozkazów” – całkowita liczba rozkazów wykonanych podczas wszystkich iteracji programu
- „Zużycie czasu” – złożoność czasowa według równomiernego kryterium wagowego
- „Zużycie czasu (Log)” – złożoność czasowa według logarytmicznego kryterium wagowego
- „Zużycie pamięci” - złożoność pamięciowa według równomiernego kryterium wagowego
- „Zużycie pamięci (Log)” - złożoność pamięciowa według logarytmicznego kryterium wagowego
- „Liczba wykonanych instrukcji” – lista instrukcji z liczbą powtórzeń podczas działania programu

Drugi element, czyli okno do generowania wykresu złożoności wymaga dłuższej analizy. Aby przejść do okna należy wybrać przycisk , który spowoduje wyświetlenie okna. Ważne, aby przed rozpoczęciem analizy wczytać program, który chcemy analizować, ponieważ podczas próby wygenerowania wykresów użytkownik otrzyma komunikat o błędzie. Na rysunku 6.25 przedstawiono wygląd okna.



Rys. 3.1 Wygląd okna do generowania wykresów złożoności

Przed przystąpieniem do generowania wykresów należy wypełnić część z wymaganych pól. Ważnymi elementami, które należy zdefiniować są wartości na taśmie wejściowej. Bez tych danych nie będzie można określić złożoności algorytmu. Ze względu jednak na dużą dowolność, jaką otrzymuje użytkownik podczas tworzenia programu, ciężko jest jednoznacznie zdefiniować rozmiar zadania przy użyciu programu napisanego w języku JavaScript. Aby uniknąć wielu błędów zdecydowano, że użytkownik sam musi zdefiniować co będzie rozmiarem zadania. W tym celu powstała lista „Ilość/Wielkość zadania” za pomocą której użytkownik określa jakie wartości znajdą się na taśmie wejściowej. Do wyboru użytkownik posiada 3 opcje:

- „Rozmiar operandu”
- „Długość taśmy wejściowej”
- „Własny ciąg cyfr”

Pierwsza opcja tj. „Rozmiar operandu” umożliwia zbadanie programu użytkownika pod kątem wzrostu wartości operandu. Do pierwszej komórki taśmy wejściowej wprowadzana jest liczba, która pozwala wywołać kolejne iteracje programu zgodnie z zaleceniami użytkownika. Opcja może być przydatna w przypadku badania programów, takich jak np. potęgowanie, obliczania silni itp. Użytkownik może zdefiniować maksymalną liczbę, jaką może wystąpić na taśmie wejściowej oraz rozdzielczość, czyli ilość danych, którą będzie badał dany algorytm. Dla przykładu podanie wartości maksymalnej 5 oraz rozdzielczości 5 spowoduje sprawdzenie algorytmu z następującymi wartościami: 1, 2, 3, 4, 5. Wartość maksymalna jest dzielona przez rozdzielczość czego wynikiem jest ciąg cyfr, które zwiększają się o stałą wartość wynikającą z

obliczeń. Na rysunku 6.26 przedstawiono wygląd okna podczas wybrania opcji „Rozmiar operandu”.

The screenshot shows a software interface with the following elements:

- A label "Ilość / Wielkość danych wejściowych:" followed by a dropdown menu currently showing "Rozmiar operandu".
- A label "Maksymalna wartość operandu / Ilość punktów na wykresie:" followed by a text input field and a dropdown menu showing "5".
- A green button labeled "Oblicz" with a right-pointing triangle icon.

Rys. 3.2 Opcje wykresu dla rozmiaru operandu

Druga z opcji, która została wymieniona dotyczy rozmiaru taśmy wejściowej. Opcja ta może być przydatna dla przypadku badania programów, które jako rozmiar zadania uwzględniają ilość wprowadzonych liczb jak np. algorytm sumowania. Użytkownik ma do wyboru kilka opcji:

- „Liczby naturalne”
- „Liczby pierwsze”
- „Liczby złożone”
- „Liczby naturalne + 0”
- „F(1) + 0”

Opcje określają jakie ciągi liczb będą reprezentowały wartości na taśmie wejściowej. Jeżeli chodzi o trzy pierwsze opcja sprawa jest prosta i nie wymaga tłumaczenia. Dla kolejnych wartości listy, gdzie występuje znak „+ 0” oznacza, że na końcu każdego ciągu dodawane jest wartość 0. Funkcja „F(1)” reprezentuje stałe wartości na taśmie wejściowej, czyli w tym przypadku wartość 1, a na końcu każdej taśmy znajduje się 0. Maksymalna wartość danych wejściowych jest reprezentowana przez listę gdzie można wybrać kilka opcji: 1, 2, 3, 4, 5, 10, 15, 20. Na rysunku 6.27 pokazano wygląd listy wyboru dla opcji „Długość taśmy wejściowej”.

Ilość / Wielkość danych wejściowych:

Długość taśmy wejściowej ▾

Max ilość danych wejściowych / Liczby :

1 ▾ Liczby naturalne ▾

Oblicz ▶

Rys. 3.3 Opcje określające wartości na taśmie wejściowej (rozmiar taśmy wejściowej)

Ostatnia z opcji, która pozwala na dowolność we wprowadzaniu liczb na taśmę wejściową i dotyczy opcji „Własny ciąg liczb”. Opcja ta pozwala na wprowadzania własnych wartości na taśmę wejściową co pozwala na badanie praktycznie wszystkich programów pod względem złożoności obliczeniowej algorytmów. Użytkownik stosując się do formatu przyjętego przez autora pracy może wprowadzić nieskończenie długi ciąg cyfr (w rzeczywistości ciąg musi być skończony ze względu na ograniczenia sprzętowe). Aby zdefiniować jeden z ciągów na taśmie wejściowej należy cyfry, które chcemy użyć, zapisać w nawiasach klamrowych. Każdy z ciągów powinien zostać rozdzielony średnikiem np. [1,2];[1,2,3];[1,2,3,4]. Natomiast wartości w klamrach muszą być cyframi. Jeżeli konwencja wprowadzania wartości zostanie zachowana emulator powinien bez problemów wygenerować wykres złożoności. W przeciwnym wypadku zostanie wyświetlony błąd, który naprowadzi użytkownika na nieprawidłową wartość. Na rysunku 6.28 przedstawiono wygląd okna wyboru dla opcji „Własny ciąg cyfr”.

Ilość / Wielkość danych wejściowych:

Własny ciąg cyfr ▾

Podaj wartości taśmy wejściowej :

Np. [0,1]; [0,2,0]; [1]; ...

Rys. 3.4 Opcja "Własny ciąg cyfr" do generowania wykresów

Po wybraniu opcji można przystąpić do generowania wykresu przy pomocy przycisku „Oblicz”. Opcje takie jak: „Wybór funkcji” oraz „Wybór złożoności” mogą zostać wybrane również po obliczeniu złożoności. „Wybór złożoności” spowoduje wyświetlenie jednego z 4 rodzaju złożoności badanych i omawianych podczas tworzenia pracy. Natomiast opcja „Wybór

funkcji” pozwala na wygenerowanie jednego z 7 wykresów w celu porównania złożoności z funkcją, która może definiować daną złożoność. Do wyboru użytkownika należą opcje:

- $O(1)$
- $O(x)$
- $O(x^2)$
- $O(x^3)$
- $O(x^4)$
- $O(\log |x|)$
- $O(x * \log |x|)$
- $O(x^2 * \log |x|)$

Dodatkowo został użyty mnożnik, który pozwala na proporcjonalny wzrost każdej z wartości funkcji o stałą wartość.