Uniwersytet w Białymstoku

Instytut Informatyki

[**Aplikacja - symulator działania komputera klasy PC w oparciu o jego układy blokowe**](https://apd.uwb.edu.pl/diplomas/55368/)

SEBASTIAN PIETROWCOW

Nr albumu: 80262

PROMOTOR:

DR INŻ. WIESŁAW PÓŁJANOWICZ

Białystok 2022r.

**SPIS TREŚCI:**

[Wstęp 2](#_Toc108095788)

[1. Historia i architektura komputerów klasy PC 4](#_Toc108095789)

[1.1 Historia komputerów osobistych 4](#_Toc108095790)

[1.2 John von Neumann 9](#_Toc108095791)

[1.3 Historia i architektura komputera IAS 11](#_Toc108095792)

[1.4 Budowa i działanie układów cyfrowych 13](#_Toc108095793)

[2. Technologie informatyczne 15](#_Toc108095794)

[2.1 Język C# 15](#_Toc108095795)

[2.2 .NET Framework 18](#_Toc108095796)

[3. Aplikacja - symulator działania komputera klasy PC w oparciu o jego układy blokowe – etapy realizacji 20](#_Toc108095797)

[3.1. Opis aplikacji 20](#_Toc108095798)

[3.1.1 Okno główne aplikacji 20](#_Toc108095799)

[3.2. Funkcjonalność aplikacji 20](#_Toc108095800)

[Komenda „cl” czyści konsole ze wszystkich wiadomości i tekstu. Najważniejszą komendą jest komenda „comp” która służy do wypisywania aktualnie wybranych komponentów symulowanego komputera. Jeśli zostanie wprowadzone tylko słowo „comp” to zostaną wypisane informacje o każdym z elementów kompótera które można wybierać. Jeśli do słowa „comp” dopiszemy odpowiednich prefix-ów to otrzymamy informacje tylko i wyłącznie o elementach przypisanych do tych prefixów. 25](#_Toc108095801)

[3.2.2. Okno edycji komputera 26](#_Toc108095802)

[3.2.3. Okno dodania nowego komponentu 29](#_Toc108095803)

[Podsumowanie 32](#_Toc108095804)

[Bibliografia 33](#_Toc108095805)

[Spis rysunków 35](#_Toc108095806)

# Wstęp

Przez wiele lat komputery osobiste ewoluowały i otrzymywaliśmy coraz bardziej wydajniejsze modele tych komputerów. W komputerach klasy PC, zaczynając od komputerów marki IBM aż po dzisiejsze urządzenia PC możemy zauważyć ich wspólną cechę. Architektura i organizacja pozostaje nie zmieniona lub w większym stopniu taka sama. W dzisiejszych czasach komputery osobiste stały się bardzo powszechne na tyle, że w prawie każdym domu możemy znaleźć przynajmniej jedno takie urządzanie. Korzystając z komputera zbytnio się nie zastanawiamy jak może działać i co się dzieje w jego wnętrzu podczas, gdy korzystamy z niego.

Celem mojej pracy licencjackiej było stworzenie aplikacji symulującej i prezentującej działanie komputera klasy PC w oparciu o jego układy blokowe. W głównej mierze skupiłem się na strukturze komputera IAS (Institute for Advanced Study) która jest architekturą klasycznego komputera. Dzięki aplikacji można lepiej poznać i zrozumieć budowę i działanie komputerów klasy PC z których bardzo często korzystamy w codziennym życiu. W pracy przeanalizowałem etapy rozwoju komputerów osobistych, opisałem wpływ rozwoju urządzeń PC na życie człowieka oraz omówiłem budowę oraz działanie układów cyfrowych występujących w architekturze aktualnych komputerów PC.

W pierwszym rozdziale przedstawiłem historię i rozwój komputerów, zaczynając do komputerów, które były wykorzystywane głównie w celach naukowych, aż do popularyzacji komputerów jako urządzeń osobistych wykorzystywanych w codziennym użytku. Opisałem również historię komputera IAS (Institute for Advanced Studies) oraz jego architekturę czyli tak zwaną architekturę von Neumanna. Ta architektura jest wykorzystana w klasycznych komputerach osobistych. Sam jej twórca, John von Neumann, miał spory wpływ na rozwój technologii komputerowej, więc w jednym z podrozdziałów przedstawiłem jego historię oraz ważniejsze dokonania szczególnie w zakresie rozwoju komputerów. Systemy w dzisiejszych komputerach w głównej mierze są zbudowane z wielu różnych układów cyfrowych takich jak na przykład bramki logiczne, liczniki, przerzutniki i tym podobne. W celu przybliżenia ich roli w systemach komputerowych oraz sposobu działania, przeznaczyłem tej tematyce podrozdział rozdziału pierwszego.

W drugim rozdziale opisałem technologie z jakich skorzystałem przy tworzeniu aplikacji symulującej zasadę działania komputera klasy PC w oparciu o jego układy blokowe wraz z wizualizacją poszczególnych elementów układu. W głównej mierze opisałem historię oraz rozwój języka w którym, została napisana aplikacja czyli języka programowania C#. Sam język przeszedł wiele zmian, z początkowych wersji, które sam autor nazywał go podobnym do Javy, aż do dzisiaj gdy C# stał się jednym z bardziej popularniejszych języków na rynku. Poza samym językiem programowania opisałem również .NET Framework z którego, korzystałem podczas tworzenia aplikacji.

Trzeci rozdział został poświęcony części praktycznej pracy, czyli aplikacji symulującej działanie komputera PC w oparciu o jego układy blokowe. Aplikacja daje możliwość ustawienia dowolnych elementów z czterech głównych typów komponentów komputera czyli procesor, karta graficzna, pamięć RAM oraz dysk twardy. Każdy z tych elementów jest pokazany w sposób graficzny oraz odpowiednio opisany. W tym rozdziale również opisałem dokładnie każde z okien aplikacji oraz ich funkcjonalności.

# 1. Historia i architektura komputerów klasy PC

## 1.1 Historia komputerów osobistych

Komputerami osobistymi możemy nazwać komputery, których cena   
i możliwości są odpowiednie do indywidualnego i prywatnego użytku. Takie komputery tworzy się do użytku codziennego dla użytkowników, którzy nie są technikami lub specjalistami w dziedzinie informatyki.



Rysunek 1. Mikrokomputer Altair 8800

Źródło: https://en.wikipedia.org/wiki/Altair\_8800

W 1974 został zbudowany pierwszy mikrokomputer Altair 8800 przez Micro Instrumentation and Telemetry System (MITS) Eda Robertsa. Odniósł on wielki sukces jako komputer przeznaczonych dla niewielkiej grupy osób. Altair był bardziej traktowany jako ciekawostka technologiczna, ale ostatecznie został sprzedany w ilości 10 tysięcy egzemplarzy.

Posiadał duży potencjał rozbudowy na przykład rozszerzenia pamięci RAM oraz możliwość podpięcia magnetofonu, nośnika pamięci masowej czy stacji dyskietek. Firma Billa Gatesa i Paula Allena uzupełniło Altaira 8800 o interpreter Microsoft BASIC, który był dialektem języka Basic. W przyszłości znany również jako Altair BASIC. Mikrokomputer od Micro Instrumentation and Telemetry System nie posiadał klawiatury i monitora. Obsługiwało się go przy pomocy przełączników na przednim panelu komputera. W przyszłych rozszerzonych wersjach była możliwość podłączenia do niego terminala szeregowego takiego jak między innymi dalekopis. Wymagało to dodatkowej karty wejścia/wyjścia. Jeden z egzemplarzy podobno pojawił się w Białym Domu. Altair 8800 od MITS jest uznawany za pierwszy komputer osobisty.

Przełomem w technologii komputerów było pojawienie się minikomputerów oraz utworzenie układów scalonych, które w sobie łączyły liczniki, rejestry, multipleksery, dekodery itp. Taką technologie scalania jednostek funkcjonalnych nazywamy średnią skalą integracji, czyli w skrócie MSI (ang. *Middle Scale Integration*). Komputery budowane   
z komponentów MSI mogły być znacznie mniejsze i tańsze, oraz były bardziej niezawodnie   
i szybsze w działaniu. Zapoczątkowanie produkcji minikomputerów wykorzystujące   
z technologie średniej skali integracji możemy przypisać firmie DEC. Firma ta produkowała maszyny znacznie mniejsze niż maszyny produkowane przez mainframe, „*jedna zgrabna szafka w odróżnieniu od wielu potężnych szaf składających się na mainframe*”[1]. Początkowo rynek minikomputerów był zdominowany przez minikomputery od firmy DEC, a zwłaszcza PDP-11, który był produkowany przez 20 lat i wciąż na niego był popyt. Minikomputery mogły być obsługiwany przez wielu użytkowników w trybie wielodostępu, kosztem znacznego zmniejszenia wydajności. Mimo takiej możliwości w praktyce najczęściej korzystało się z komputera w modelu „*używania jednego komputera przez jednego użytkownika do jednego celu*” [1]. Było to preludium do powstania komputerów osobistych czyli komputerów PC (*Personal Computer*).

Mikrokomputerem którym zapoczątkowano nową epokę, był Apple, stworzony przez Stevena Paula Jobsa oraz Stephana Gary-ego „Steve” Wozniaka w roku 1976. Apple przeznaczony był głównie dla hobbystów technologicznych. Posiadał sześćdziesiąt różnych układów scalonych. Aby móc korzystać w pełni z Apple użytkownik musiał dodać zasilacz, obudowę oraz peryferia takie jak klawiatura i monitor. Jako pierwszy powszechny komputer korzystał z klawiatury i monitora.

Największą sławę wśród komputerów osobistych zdobył Macintosh. Jako pierwszy posiadał graficzny interfejs użytkownika oraz myszkę jako podstawowe peryferie, co było w tamtych czasach czymś nowym. Mikrokomputer od Apple były urządzeniami ekskluzywnymi, skierowane do bardziej zamożnych i wymagających użytkowników, często artystów.



Rysunek 2. Komputer IBM PC

Źródło: https://www.geekweb.pl/inne/kartka-z-kalendarza/item/412-39-lat-temu

IBM widząc rosnącą popularność komputerów od Apple, odniósł się do tej sytuacji powołując nowy zespół, którego zadaniem było utworzenie nowego rewolucyjnego komputera osobistego. Na czele nowo powstałego zespołu stanął Philip Don Estridge’a z IBM Entry System Division w Boca Raton na Florydzie. W dość krótkim czasie, zespół zbudował mikrokomputer, który nazwano IBM PC 5150, a jego wbudowany monochromatyczny monitor był nazwany IBM 5151. IBM PC został zaprezentowany 12 sierpnia 1981 roku. Przez pośpiech komputer od IBM odstawał od konkurencji, które słynęły z zadbanych i pięknych obudów. IBM PC wyglądał jakby „*IBM całkowicie zrezygnował ze wsparcia artystów plastyków zajmujących się wzornictwem przemysłowym – po prostu inżynierowie zmontowali do kupy wszystkie potrzebne urządzenia, a potem przyszedł ślusarz i osłonił całość niedbale wygiętą blachą. Coś w tym jest – nie da się ukryć…*” [1]. Oprócz braku dbałości o estetykę firma IBM zaskoczyła czymś wyjątkowym i rzadko spotykanym: udostępniła pełną dokumentacje IBM PC oraz pozwoliła niezależnym firmom tworzyć kopie swojej maszyny. Powstał wielki „boom”, niezliczone ilości firm zaczęły wytwarzać i sprzedawać komputery korzystając z dokumentacji technicznej IBM PC 5150. Takie kopie określano jako kompatybilne z IBM PC, co oznacza że kopia jest w pełni zgodna z oryginałem.

Pod względem wykonania „kompatybilne” kopie komputera IBM bardzo różniły się od siebie przez co firmy, głównie z Azji, produkowały znacznie tańsze komputery IBM PC. Wiele takich maszyn trafiło również do Polski. Zaczął się trend na wykonywaniu na nich prac informatycznych w zakładach pracy i instytucjach naukowych. Na o wiele większą skale niż poprzednie komputery osobiste, komputer IBM trafił do domów prywatnych. Do końca 1985 statystycznie 90% wykorzystywanych komputerów PC to były kopie IBM PC.

Przez wielki napływ klonów IBM PC, popyt na oryginalny produkt firmy IBM drastycznie spadło, przez co gigant zaczął zauważać zagrożenie finansowe związanym ze swoim produktem. Żeby rozwiązać problem, w 1983 wypuścił nowy produkt który stał się hitem na rynku: IBM PC/XT. Sam skrót XT pochodzi od angielskiego słowa extended czyli rozszerzony/poszerzony. Tańsza wersja IBM PC/XT posiadała stacje do zapisu i odczytu danych na dyskietkach, za to niektóre droższe wersje były wyposażone w dyski twarde o pojemności 10 MB, następnie rozszerzono do 20 MB. Sporą popularnością mogła pochwalić się wzbogacona wersja „turbo” z zegarem o częstotliwości 10 Mhz oraz wersja XT289 z udoskonalonym mikroprocesorem Intel 80286 . Podstawowe XT posiadało procesor Intel 8088 oraz zegar o częstotliwości 4,77 MHz.

Po wielkim sukcesie rynkowym IBM PC/XT, wprowadzono na rynek IBM PC/AT w sierpniu 1984. Nowa wersja szybko wyparła poprzedników. Było to spowodowane głównie przez lepszy mikroprocesor Intel 80286, szybszy zegar oraz dyski twarde o większej pamięci. Następnie pojawiły się coraz to nowsze modele związane z coraz to nowszymi mikroprocesorami Intel 80386, Intel 80486 oraz Intel Pentium.

Obok markowych komputerów od IBM pojawiły się komputery IBM PC/XT od firmy Dell Computer Corporation. Nie wprowadzono żadnych własnych innowacji technicznych ale Dell produkował maszyny o której mawiano, że „są bardziej IBM niż produkty IBM” [1]. Firma Dell istnieje do dziś i słynie ze swej wysokiej jakości i niezawodności. Samą jakość komputerów oceniano na podstawie ich szybkości działania. Do testowania komputerów wykorzystywano specjalne oprogramowanie do sprawdzenia szybkości wykonywania zadań przez komputer. Najsłynniejszym takim programem był SpeedTest od firmy Landmark. SpeedTest pozwalał m.in. na określenie współczynnika szybkości działania procesora, szybkość wykonywania operacji graficznych.

Rozwój pierwszych komputerów był ściśle związany z rozwojem amerykańskiej informatyki i architektury mikrokomputerów. Nie oznacza to, że w innych krajach nie powstawały również ciekawe komputery. Przykładowo, w latach pięćdziesiątych w ZSRR zaczęto produkować pierwsze komputery m.in. BESM który wykorzystywano do działań związanymi z lotem w kosmos oraz MESM który pierwszy raz został uruchomiony 25 grudnia 1951. Jednym z ważnych wydarzeń historycznych w historii komputerów było powstanie pierwszego polskiego komputera zwanego XYZ w roku 1958 przez Zakład Aparatów Matematycznych PAN. Tworzeniem XYZ kierował prof. Leon Łukaszewicz. Maszyna stworzona głównie w charakterze naukowym i doświadczalnym, ale dzięki zebranym informacjom pozyskanym przy budowie i pracy komputera zbudowano pierwszy polski komputer użytkowy zwany ZAM-2. Komputer ten był wyprodukowany na potrzeby rynku krajowego i na eksport w ograniczonej ilości 12 sztuk. ZAM-2 był przeznaczony do wykonywania obliczeń praktycznych m.in. dla Towarzystwa Ubezpieczeń WARTA.

Jednym z kluczowych elementów każdego komputera jest pamięć, szczególnie pamięć operacyjna. Do końca lat 70 korzystano z pamięci ferrytowej w postaci magnetycznych pierścieniowych rdzeni z przeplecionymi przewodami. Zostały wynalezione przez Terence Williams w 1953 roku. Cechowała je powolna praca i mała pojemność.

W roku 1975 zostały zastąpione przez do dziś używaną pamięć RAM (ang. *Random Acces Memory*), która były budowana z układów scalonych. RAM jest pamięcią tymczasową co oznacza, że po wyłączeniu urządzenia lub po zaniku napięcia cała pamięć zostaje utracona. Pamięć ta służy do przechowywania informacji i danych potrzebnych do działania systemu. Do działania systemu potrzebujemy również pamięci stałej która w przeciwieństwie do pamięci tymczasowe RAM nie utraci danych po zaniku napięcia lub wyłączeniu urządzenia. Dobrym przykładem jest do dziś używana pamięć ROM (ang. *Read Only Memory*) w której na stałe zostają zapisane informacje wprowadzone przez producenta komputera. Dzięki odporności na spadki napięcia nie czyściła się przy zaniku napięcia lub wyłączeniu maszyny. Pamięć ta zazwyczaj zawierała stałe części systemu operacyjnego.

## 1.2 John von Neumann

John von Neuman urodził się 28 grudnia 1903 w Budapeszcie w rodzinie żydowskiej. Od najmłodszych lat przejawiał zainteresowanie naukami ścisłymi oraz posiadał nadzwyczajnie dobrą pamięć, ”która pozwalała mu po krótkim spojrzeniu na stronę książki, cytować dokładnie jej zawartość „ [6]. Już w wieku 6 lat potrafił dzielić liczby ośmiocyfrowe. Jego prawdziwym nazwiskiem jest János Lajos Neumann. Podczas pobytu w Niemczech przyjął nazwisko Johann von Neumann, ale dziś jest znany pod swym amerykańskim imieniem John von Neumann. Ukończył studia na Uniwersytecie Berlińskim na kierunku matematyka oraz na kierunku chemia. Ukończył również studia na uniwersytecie w Hamburgu oraz w Technische Hochschule w Zurychu.



Rysunek 3. John von Neumann

Źródło: https://pl.wikipedia.org/wiki/John\_von\_Neumann

W roku 1930 otrzymał zaproszenie na poprowadzenie cyklu wykładów na Uniwersytecie Princeton. Po wyjeździe do Stanów Zjednoczonych Ameryki postanowił pozostać w niej aż do końca życia. Przez prawie dwadzieścia lat, aż do końca lat 30 publikował ważne artykuły naukowe dzięki, którymi zdobył międzynarodową reputację. Był starszym członkiem jednego z ważniejszych ośrodków badawczych na świecie, czyli Institute for Advanced Study. Przed wybuchem wojny poświęcał wiele czasu technologii hydrodynamiki wzajemnego oddziaływania fal uderzeniowych ważnych między innymi dla wojska. Badania nad tą technologią kosztowały wiele godzin obliczeń przez co von Neumann zainteresował się maszynami matematycznymi. Podczas wojny został zatrudniony w armii amerykańskiej jako konsultant do rozwiązywania problemów badawczych. Potrafił rozwiązać skomplikowany problem w mgnieniu oka przez co był bardzo ceniony w armii. Podczas swojej pracy w IAS powierzono mu projekt zbudowania nowoczesnego komputera, którego celem było wykonywanie obliczeń związanych z bronią atomową.

W ekonomii próbował skonstruować „*matematyczny model równowagi ekonomicznej dla równomiernie rozwijającej się gospodarki zamkniętej o niegranicznych zasobami i stałymi zwrotami do skali w produkcji*” [3]. Model po raz pierwszy został zaprezentowany w 1932 na Uniwersytecie Princeton, oraz został zaprezentowany na wykładzie Karla Mangera w 1937 w Wiedniu, dzięki czemu zyskał na popularność. John von Neumann udowodnił również twierdzenie minima, które mówi ona że „*w grze o sumie zerowej z pełną informacją istnieje strategia, która pozwala każdemu graczowi na zminimalizowanie ich maksymalnych strat. Rozważając każdą możliwą strategię gracz musi wziąć pod uwagę wszystkie możliwe odpowiedzi drugiego gracza i maksymalną stratę. Wtedy może zastosować strategię pozwalającą na minimalizację największych strat*” [3].

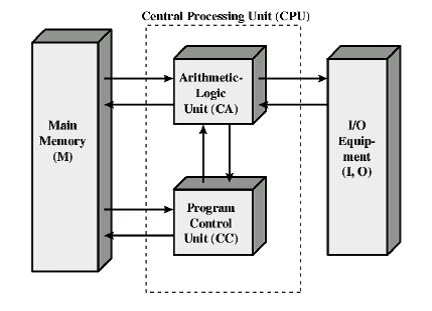
Ambicjami Johna von Neumanna było utworzenie superszybkiej maszyny liczącej oraz sztucznej inteligencji. Promował używanie maszyn matematycznych do rozwiązywania problemów matematycznych których człowiek jeszcze nie zdołał rozwiązać. Również ulepszył i zmodyfikował maszynę ENIAC, którą uważaną za pierwszy komputer elektroniczny na świecie, aż do odtajnienia brytyjskich danych. W 1945 roku wraz z Johnem Presperem Eckertem i Johnem Williamem Mauchlym stworzył projekt komputera z oprogramowaniem i danymi zapisywanymi w pamięci. Projekt ten był znany w świecie jako architektura von Neumanna, przez co współpracownicy von Neumanna mieli do niego pretensje. Przyczyniło się to do rozpoczęcia pracy Johna von Neumanna w 1946 roku na rzecz IAS mimo trwającej pracy nad EDVAC. Praca w IAS poskutkowała stworzeniem projektu komputera uniwersalnego i taniego w produkcji, na bazie architektury von Neumanna. Komputer ten zwano komputerem IAS oraz komputerem von Neumanna.

## 1.3 Historia i architektura komputera IAS

Idea koncepcji przechowywania programów w pamięci była przypisywana projektom ENIAC-a, który pobierał rozkazy odczytując je z pamięci wraz z danymi. Programy mogły być instalowane lub zmieniane przez zmiany w pamięci. Koncepcje ta była przypisywana Johnowi von Neumannowi, mimo że w tym samym czasie podobną koncepcje opracował Alan Mathison Turing.

W 1946 w Princeton Institute for Advanced Studies ruszyły prace nad komputerem wykorzystującym programy zapisane w pamięci. Nad projektem pracował między innymi wyżej wspomniany von Neumann. Komputer ten nazywano skrótem IAS. Jego budowa trwała aż do 1952, ale jego architekturę wykorzystano we wszystkich kolejnych komputerach przeznaczonych do ogólnego użytku.

Architektura komputera IAS składała się z czterech składników: pamięci głównej która odpowiada za przechowywanie danych i programów, które są odczytywane przez komputer, jednostki arytmetyczno-logiczne (ALU) która pozwalała na działania na danych binarnych, jednostki sterującej która interpretowała rozkazy z pamięci i powodowała ich wykonanie oraz urządzeń wejścia-wyjścia którymi kierowała jednostka sterująca.



Rysunek 4: Struktura komputera IAS

Źródło: Computer Organization And Architecture Designing For Performance Eighth Edition

Możemy zauważyć że komputer najczęściej będzie wykonywał operacje arytmetyczne takie jak dodawanie, odejmowanie, dzielenie i mnożenie, przez co rozsądnym by było utworzenie części która będzie zajmowała się wykonywaniem tych operacji, co oznacza wystąpienie jednostki arytmetyczno-logicznej w tej strukturze. Ta jednostka zawiera w sobie m.in. rejestr buforowy pamięci (MBR) który zawiera słowo, które ma być przechowywane w pamięci, lub też jest wykorzystywany do pobierania słów z pamięci.

Komputer również potrzebuje jednostki sterującej, która będzie zarządzała przepływem danych oraz będzie szeregować jego operacje. „*Jeśli urządzenie ma być elastyczne, to znaczy możliwie uniwersalne, należy rozróżniać specyficzne rozkazy związane z określonym problemem i ogólne „organy" sterujące dbające o wykonanie tych rozkazów*”. [9] Element który zrealizuje te zadania nazywamy jednostką sterującą.

Jednostka sterująca maszyny IAS przy uruchomieniu pobiera rozkaz z pamięci i następnie go wykonuje. Może wykonywać tylko jeden rozkaz w tym samym czasie. Jednostka centralna składa się między innymi z rejestru adresowego pamięci (MAR), który określa adres słowa w pamięci w celu zapisania go w rejestrze buforowym pamięci lub odczytania z niego, rejestru rozkazów (IR) który zawiera w sobie „*8-bitowy kod operacyjny rozkazu, który jest wykonywany”* [9]*, buforowego rejestru rozkazów (MBR) który wykorzystuje się do przechowywania rozkazu pochodzącego ze słowa oraz licznik programu (PC), który „zawiera adres następnej pary rozkazów, która ma być pobrana z pamięci*” [9].

Żeby komputer mógł wykonywać długie i skomplikowane programy, musi mieć do tego pamięć w której będzie mógł zapisać elementy które musi zapamiętać podczas wykonywania programu. Do tego jest potrzebna pamięć główna. Pamięć komputerze IAS miała 1024 słowa z których każde zawiera 40 cyfr binarnych (bitów). Przechowywano tam dane jak i również rozkazy. Taka budowa pamięci wymuszała prezentowanie liczb w formie binarnej, a każdy rozkaz musiał być napisany również w kodzie binarnym.

Komputer IAS działa powtarzalnie wykonując cykl rozkazu. „*Każdy cykl rozkazu składa się z dwóch podcykli. Podczas cyklu pobrania rozkazu kod operacji następnego rozkazu jest ładowany do rejestru rozkazu, natomiast część adresowa - do rejestru MAR*.” [9]. Gdy kod programu znajdzie się w rejestrze rozkazów to, wykonywany jest cykl wykonywania, podczas którego układy sterujące wykonują rozkaz, wysyłając odpowiedni sygnały, które powodują że dane są przenoszone albo ALU wykonuje operacje. Architektura dzisiejszych komputerów w większości dalej opiera się na strukturze komputera IAS.

## 1.4 Budowa i działanie układów cyfrowych

Jeśli przyjrzymy się dokładniej budowie systemu komputerowego, można zauważyć, że system składa się z wielu układów cyfrowych, których budowa opiera się o bramki logiczne. Bramki logiczne mają za zadanie obsłużyć operacje logiczne takie jak:

* iloczyn logiczny obsługiwany przez bramki AND i NAND, gdzie AND zwraca wartość prawdziwą wtedy i tylko wtedy gdy oba jej wejścia będą miały wartość prawdziwą. Bramka NAND jest negacją AND i zwraca wartość fałszywą wtedy i tylko wtedy gdy oba wejścia są prawdziwe,
* suma logiczna obsługiwana przez bramki OR i NOR, w których bramka OR przyjmuje wartość prawdziwą wtedy gdy któryś z wejść jest prawdziwe lub gdy oba są prawdziwe. Bramka NOR jest zaprzeczeniem bramki OR i zwraca wartość prawdziwą wtedy i tylko wtedy gdy oba wejścia przyjmują wartość fałszywą,
* negacja obsługiwana przez bramkę NOT, która w przeciwieństwu do innych bramek posiada tylko jedno wejście i zwraca wartość prawdziwą, gdy na wejściu otrzymuje wartość fałszywą, ale gdy na wejściu otrzyma wartość prawdziwą, zwraca fałsz,
* różnica symetryczna obsługiwana przez bramkę XOR, która przyjmuje wartość prawdziwą wtedy i tylko wtedy, gdy tylko jedno z wejść ma wartość prawdziwą, w przeciwnym wypadku bramka zwróci wartość fałszywą.

Układy cyfrowe klasyfikujemy na dwie główne klasy, czyli układy kombinacyjne do których zaliczają się między innymi komutatory i konwertery kodów oraz układy sekwencyjne, do którego należą między innymi przerzutniki i liczniki.

Układy kombinacyjne są układami w których stan wyjścia zależy tylko i wyłącznie od stanu wejścia. Stany wejścia i wyjścia są funkcjami boolowskimi, które przyjmują ale wartość prawdziwą true albo wartość fałszywą false. W tym rodzaju układów nie występuje zjawisko sprzężenia zwrotnego czyli wpływu sygnału wyjściowego na sygnał wejściowy. Przykładem układu kombinacyjnego jest komutator multiplekser, który służy do wybierania jednego sygnału z kilku sygnałów wejściowych, żeby następnie przekazać go do wyjścia. Jednymi z ważniejszych układów kombinacyjnych są konwertery kodów czyli na przykład, koder którego zadaniem jest zamiana kodu „1 z n” na kod binarny oraz dekoder, który odczytuje kod binarny i zamienia go na kod „1 z n”.

Drugim rodzajem układów cyfrowych są układy sekwencyjne, które charakteryzują się tym, że sygnał przyjmowany na wejściu jest zależny od sygnału otrzymywanego na wyjściu oraz od poprzedniego sygnału nazywanego stanem wewnętrznym, który jest zapamiętany w zespole rejestrów. Jeśli sygnał zapisany w pamięci nie ulegnie zmianie pod wpływem sygnału wejściowego to nazywamy go stanem stabilnym.

Wśród układów sekwencyjnych rozróżniamy dwa rodzaje. Pierwszy czyli układ asynchroniczny, w którym „*zmiana sygnałów wejściowych natychmiast powoduje zmianę wyjść. W związku z tym układy te są szybkie, ale jednocześnie podatne na zjawisko hazardu*” [12]. Zjawisko hazardu jest niekorzystnym zjawiskiem w układach, który polega na tym, że w stanach przejściowych powstają błędne stany które są wysyłane do wyjścia. Drugim rodzajem układów sekwencyjnych są układy synchroniczne w których, zmiana stanu wewnętrznego występuje tylko i wyłącznie w określonym momencie który jest określany przez sygnał zegarowy. Taki rodzaj układu charakteryzuje się tym, że „nawet gdy stan wejść się nie zmienia, to stan wewnętrzny – w kolejnych taktach zegara – może ulec zmianie.” [12 ]. Jeśli układ odpowiada na określony stan zegar to nazywamy go układem statycznym, ale jeśli układ reaguje na zmianę sygnału zegarowego to nazywamy go układem dynamicznym. Gdy układ sekwencyjny synchroniczny nie ma wyjścia, a jedyną rzeczą która go charakteryzuje jest stan wewnętrzny, to w takim wypadku możemy go nazwać układem autonomicznym. Takie układy są stosowane w zegarkach elektryczny.

Z przykładów układów sekwencyjnych możemy wspomnieć o przerzutnikach, które stosuje się do przechowywania danych do których jest potrzebny ciągły dostęp. Ze względu na łatwy dostęp do danych w celu odczytu lub zapisu, przerzutniki stosuje się do zapamiętania stanu układu, przy implementacji liczników i rejestrów przesuwających. Innym przykładem układu sekwencyjnego są liczniki, które wykorzystuje się do zliczania wystąpień sygnału zegarowego. Tworzy się je przy wykorzystaniu kilku przerzutników.

Układy sekwencyjne wykorzystuje się również przy zapisie danych. Przykładem tego jest rejestr, który zapamiętuje informacje bitowe i przetwarza zapamiętane informacje. Rejestry są zbudowane z przerzutników oraz z sieci logicznych, które przetwarzają bity przechowywane w rejestrze. Można zatem takie układy traktować jako pamięć podręczną.

# 2. Technologie informatyczne

## 2.1 Język C#

Do stworzenia aplikacji wykorzystano język programowania C# który został zaprojektowany w latach 1998 – 2001 pod kierownictwem Andersa Hejlsberga dla firmy Microsoft. Celem projektu było stworzenie języka który jest „*prostym, nowoczesnym, zorientowanym na obiekty językiem ogólnego przeznaczenia”* [13]. W tamtych czasach można było zauważyć że język Java osiągnął wyżej wymienione cele projektowe.

Wersja 1.0 języka została wydana w roku 2002 wraz z Visual Studio .NET 2002 i zauważono że wygląda podobnie jak język Java. C# 1.0 nie był jeszcze kompletny co można było zauważyć po braku „*wbudowanych funkcji asynchronicznych i niektórych funkcji slick wokół typów ogólnych, które zostały przyznane. W rzeczywistości brakowało mu w ogóle rodzajów ogólnych*” [14]. Najważniejszymi funkcjami języka C# w wersji 1.0 były między innymi: struktury, zdarzenia, Operatory i wyrażenia, klasy, interfejsy, atrybuty, instrukcje. Wersja 1.0 w porównaniu do dzisiaj wyglądała na ubogą i pozbawioną funkcji.

W roku 2005, firma Microsoft przedstawiła, wraz z prezentacją .NET Framework 2.0, rozszerzenie do języka C# nazwane C# 2.0. Rozszerzenie wprowadzało między innymi: typy ogólne, typy częściowe, typy wartości dopuszczające wartość null, metody anonimowe, iteratory, klasy statyczne i konwersje grup metod. „*Chociaż język C# mógł zostać uruchomiony jako ogólny język Object-Oriented (OO), język C# w wersji 2.0 zmienił się w pośpiechu. Gdy mieli nogi pod nimi, poszli po kilku poważnych punktach bólu dewelopera. I poszli po nich w znaczący sposób*” [13]. Wprowadzenie metod i typów pozwalało pracować na dowolnych typach danych, zachowując przy tym bezpieczeństwo typu. Wprowadzenie iteratorów umożliwiło wykorzystanie pętli foreach, aby uzyskać dostęp do każdego elementu listy. Iteratory zwiększyły czytelność kodu oraz sprawiły że język C# był łatwiejszy do odczytania i zrozumienia.

Pod koniec roku 2007 zaprezentowano C# 3.0 oraz Visual studio 2008. Ta wersja wprowadziła ważne zmiany w rozwoju C#. Ta wersja wyprowadziła C# z cienia Javy. Ważniejszymi funkcjami wprowadzonymi do języka C# w wersji 3.0 między innymi są:

* Typy anonimowe które pozwalały na hermetyzacje „*zestawu właściwości tylko do odczytu do pojedynczego obiektu bez konieczności jawnego definiowania typu*” [13]. Można je utworzyć czy użyciu operatora new i inicjatora obiektu.
* Właściwości generowane automatycznie co oznacz, że gdy akcesorach właściwości nie są wymagane nadprogramowe logiki, implementowane właściwości automatycznie tworzą zwięzłości deklaracji.
* Metody rozszerzeń które umożliwiają dodawania do istniejącej klasy nowych metod bez ich dziedziczenia. Metody te są statyczne, ale wywołuje się je jak metodami rozszerzonymi. W kodzie nie ma zbyt wyraźnej różnicy między wywołaniem metody zdefiniowanymi w typie a metodami rozszerzonymi.
* Zmienne lokalne o typach niejawnych, które stosuje się używając słowa var. Takie typy stosuje się przy wyrażeniach wywołujących konstruktor. Funkcja var pozwala nie powtarzać nazwy typu przy deklarowaniu zmiennej i obiektu.

Na kolejną wersje trzeba było czekać 3 lata, ale w roku 2010 wydano C# 4.0. W tej wersji dodano słowo kluczowe dynamic dzięki które umożliwiało tworzyć i obsługiwać konstrukcje dynamiczne podobne do tych z języków typizowanych dynamicznie taki jak na przykład JavaScript. Oprócz obsługi języków dynamicznych wprowadzono również: kontrawariancja i kowariancje oraz argumenty opcjonalne i nazwane.

W roku 2012 ukazał się C# 5.0, który skupił się tylko na funkcjonalności asynchronicznej języka, wprowadzając przy tym słowa kluczowe await oraz async, które pomagają przy programowaniu funkcji asynchronicznych. Dużo większe zmiany wprowadził C# 6.0. W przeciwieństwie do wersji 3.0 i 5.0 które skupiały się nad rozwinięciem programowania obiektowego języka C#, wersja 6.0 skupia się nad dodaniem wielu mniejszych funkcji, które mają za zadanie zwiększyć wydajność programowania w tym języku. Jednymi z ważniejszych wprowadzonych zmian i funkcji były m.in.:

* Filtr wyjątków when w kontekście catch dzięki któremu można było określić warunek który musi być spełniony, aby wyjątek mógł zostać wykonany
* Nowy operator nameof, który zwraca nazwę typu, elementu członkowskiego lub zmiennej jako stałą ciągu np. Console.WriteLine(nameof(List<int>).
* Statyczny import przy użyciu using pozwalający na importowanie m.in. bibliotek.
* Umożliwienie użycia słów async i await w blokach catch/finally

W wersji C# 7.0 można było zauważyć spowolnienie rozwoju języka głównie przez to że ta wersja oraz wersje od 7.1 do 7.3 wprowadzały głównie małe funkcje i zmiany w języku oraz udoskonalały kompilator języka c# głównie opcjami -refout i -refonly.

Jedną z ważniejszych wersji języka była wersja 8.0 wydana w 2019 roku. Uważa się, że 8.0 jest to „*pierwsza główna wersja języka C#, która jest przeznaczona dla platformy .NET Core*” [13]. Ta wersja wprowadziła bardzo dużo zmian i udogodnień m.in. takich jak:

* Wprowadzenie nowego operatora przypisania „?=”, który przypisuje „*wartość swojego operandu po prawej stronie do jego operandu po lewej stronie tylko wtedy, gdy operand po lewej stronie ma wartość null*” [13].
* Dodanie modyfikatora „readonly”, którego można zastosować przy zmiennych, funkcjach i elementach członkowskich struktury. Wskazuje że dany element nie można w żaden sposób edytować i jest przeznaczony tylko do odczytu.
* Typy referencyjne dopuszczające wartość null. Taki typ ustawia się dopisując „?” przy nazwie typu np. string?.
* Strumienie asynchroniczne z, których żeby skorzystać wystarczy dopisać przed pętlą foreach słówka await, dzięki czemu pętla foreach wykona się asynchronicznie.

Na moment pisania tej pracy, najnowszą wersją języka C# jest wersja 10. Ta wersja jest obsługiwana przez Visual Studio 2022, a kompilator tej wersji jest częścią platformy .NET 6.0. W tej wersji usprawniono C# oraz dodano funkcje m.in takie jak:

* Dodano modifikator global która sprawia, że kompilator uznaje dyrektywę z tym słowem jako globalną czyli taką, która ma zastosowani we wszystkich plikach źródłowych podczas kompilacji, czyli zazwyczaj wszystkie pliki projektu.
* Ulepszono wyrażenie lambda. Od tej wersji „*Wyrażenia lambda mogą mieć typ naturalny, w którym kompilator może wywnioskować typ delegata z wyrażenia lambda lub grupy metod.*” [13].
* Zdjęto ograniczenia w dekonstrukcji które uniemożliwiało przypisanie istniejącej zmienne i zadeklarowanie innej nowej zmiennej w tym samym dekonstruktorze.

## 2.2 .NET Framework

.NET Framework jest to wieloplatformowa platforma programistyczna open source służąca do tworzenia aplikacji. Framework ten nie jest ściśle związany z żadnym jednym językiem programowania, a można go wykorzystać przy aplikacjach pisanych w językach: C#, C++/CLI, Delphi 8, J#, F# oraz Visual Basic .NET. Platforma .NET opiera się na „środowisku uruchomieniowym o wysokiej wydajności, które jest używane w środowisku produkcyjnym przez wiele aplikacji o dużej skali” [15]. Każda wersja .NET Framework posiada bibliotekę klas bazowych, biblioteki zarządzania i środowisko uruchomienia języka CLR. Biblioteka klas bazowych to obiektowa kolekcja często używanych typów przeznaczona do tworzenia aplikacji. Przykładem takiej biblioteki jest technologia windows forms lub oferowane przez ASP.NET, technologia web forms. Środowisko uruchomienia pomaga w zarządzaniu kodem w czasie wykonywanie oraz zapewnia usługi takie jak zarządzanie pamięcią, komunikacja zdalna oraz zarządzanie wątkami. „.NET Framework mogą być hostowane przez niezarządzane składniki, które ładują środowisko uruchomieniowe języka wspólnego do swoich procesów i inicjują wykonywanie kodu zarządzanego, tworząc w ten sposób środowisko oprogramowania wykorzystujące funkcje zarządzane i niezarządzane. .NET Framework nie tylko udostępnia kilka hostów środowiska uruchomieniowego, ale także obsługuje tworzenie hostów środowiska uruchomieniowego innych firm.” [15]. Przy pomocy .NET można tworzyć aplikacje internetowe, jak również aplikacje działające na systemach, na których istnieje implementacja tej platformy programistycznej. Pierwsza wersja .NET została wydana w roku 2002 a wersje oznaczono jako 1.0. Wraz z pierwszą wersją udostępniono darmowe narzędzie programistyczne zwane Visual Studio .NET. Kolejne wersje .NET framework pojawiały się w odstępach od roku do dwóch lat i wraz z większością nowych wersji były udostępniane kolejne wersje narzędzia Visual studio. Od roku 2005 narzędzie zmieniło lekko nazwę przez co z nazwy zniknęło słowo .NET i każda kolejna wersja wraz z tą z roku 2005 nazywała się na przykładzie wyżej wspomnianego roku Visual Studio 2005, gdzie rok po nazwie oznacza rok wydania i tylko on się zmieniał w kolejnych latach. W roku 2014 Microsoft zmienił zasady udostępniania .NET Framework na open source, a w 2019 wydał wersje 4.8, która jest ostatnią wersją .NET Framework i będzie dalej wspierania.

# 3. Aplikacja - symulator działania komputera klasy PC w oparciu o jego układy blokowe – etapy realizacji

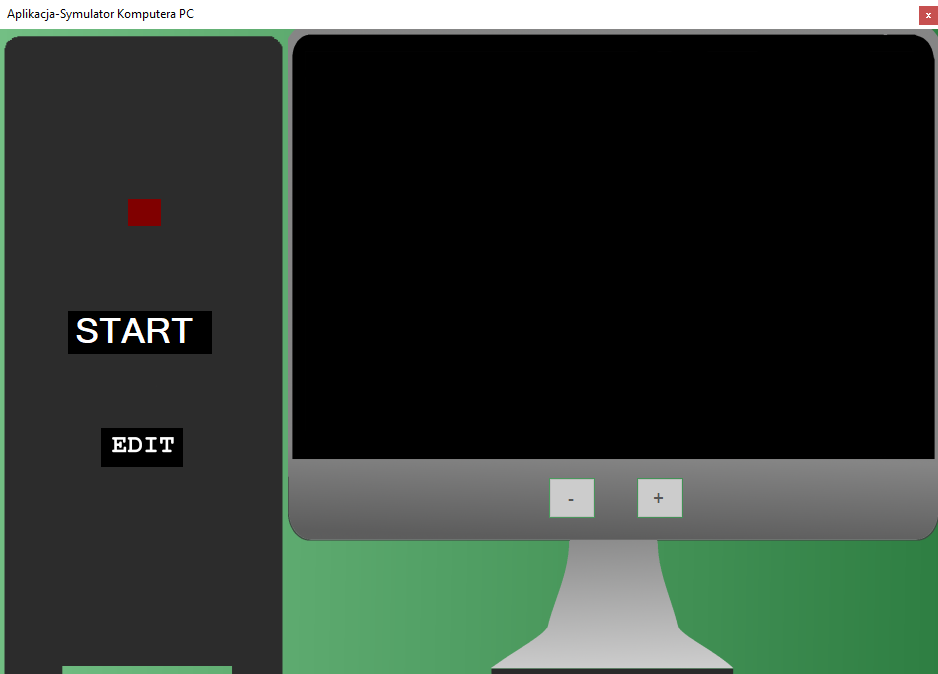
## 3.1. Opis aplikacji

Stworzona przeze mnie aplikacja symuluje działanie komputera klasy PC   
w oparciu o jego komponenty blokowe. Symulator pozwala w graficzny sposób wybrać poszczególne komponenty komputera PC. W oknie startowym znajduje się monitor   
z przyciskiem „+” i „-” oraz jednostka centralna komputera na której, znajdują się dwa przyciski oraz dioda informująca o stanie działania komputera. Jeśli użytkownik podczas korzystania z symulowanego komputera użyje, któregoś z przycisków na monitorze to odpowiednio do symbolu na przycisku, powiększają lub pomniejszają czcionkę konsoli. Po wybraniu przycisku odpowiadającego za edycje komponentów komputera, otworzy się okno edycji w którym można wybrać lub dodać komponent komputera. Po wybraniu komponentów i podłączeniu elementów wejścia/wyjścia, można w pełni korzystać   
z symulowanego ekranu komputera w oknie głównym w którym widać konsolę do której, jest możliwość wpisywania komend tekstowych.

Dzięki odpowiedniej komendzie można wypisać specyfikacje wybranych przez użytkownika komponentów dzięki czemu użytkownik widzi, że wybrane komponenty rzeczywiście znalazły się w symulowanym komputerze. Użytkownik przy pomocy odpowiedniej komendy, może również wypisać odpowiedni komponent

## 3.2 Funkcjonalności aplikacji

### 3.2.1 Okno główne aplikacji

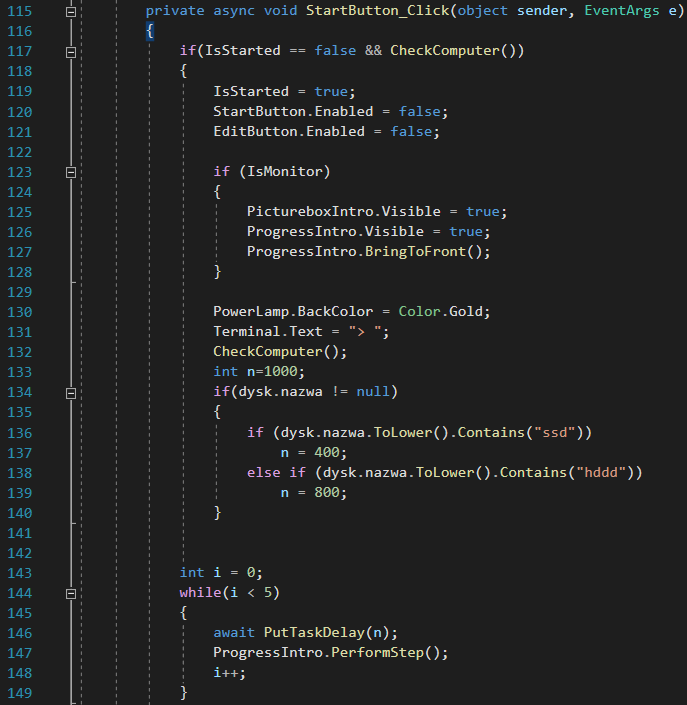


Rysunek 5. Okno główne aplikacji

Źródło: opracowanie własne

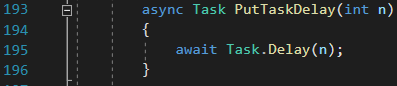
Po uruchomieniu aplikacji, pojawia się okno główne w którym znajduje się ekran monitora oraz jednostka centralna. Oba urządzenia na samym starcie są wyłączone.

Na jednostce centralnej znajduje się dioda, która informuje o stanie pracy komputera czyli przy wyłączonym komputerze dioda czerwona, przy procesie włączania dioda zmienia kolor na żółty, a po włączeniu się komputera dioda zielona. Przycisk z napisem „START” uruchomi komputer pod warunkiem, że wszystkie komponenty są zamontowane. Za uruchomienie komputera odpowiada funkcja StartButton\_Click, która jest funkcją prywatną oraz asynchroniczną w celu zablokowania przycisków podczas włączania i ponownego ich odblokowania po odpowiednim czasie gdy komputer zostanie włączony. Na początku funkcja sprawdza czy komputer jest wyłączony oraz czy ma wszystkie komponenty. Jeśli oba warunki są spełnione to funkcja uruchomia proces włączania. Gdy monitor jest podłączony to wtedy na monitorze wyświetli się ekran uruchamiania.



Rysunek 6. Fragment funkcji StartButton\_Click odpowiadający za proces włączania komputera Źródło: opracowanie własne

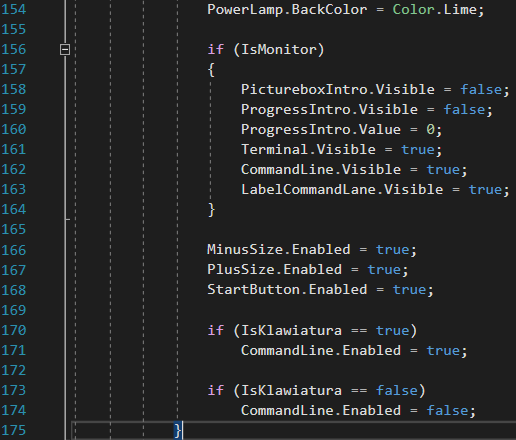
Do odczekania odpowiedniej ilości sekund funkcja korzysta z funkcji asynchronicznej PutTaskDelay, dzięki której podczas włączania nie jest wstrzymywany cały wątek aplikacji.



Rysunek 7: Funkcja PutTaskDelay

Źródło: opracowanie własne

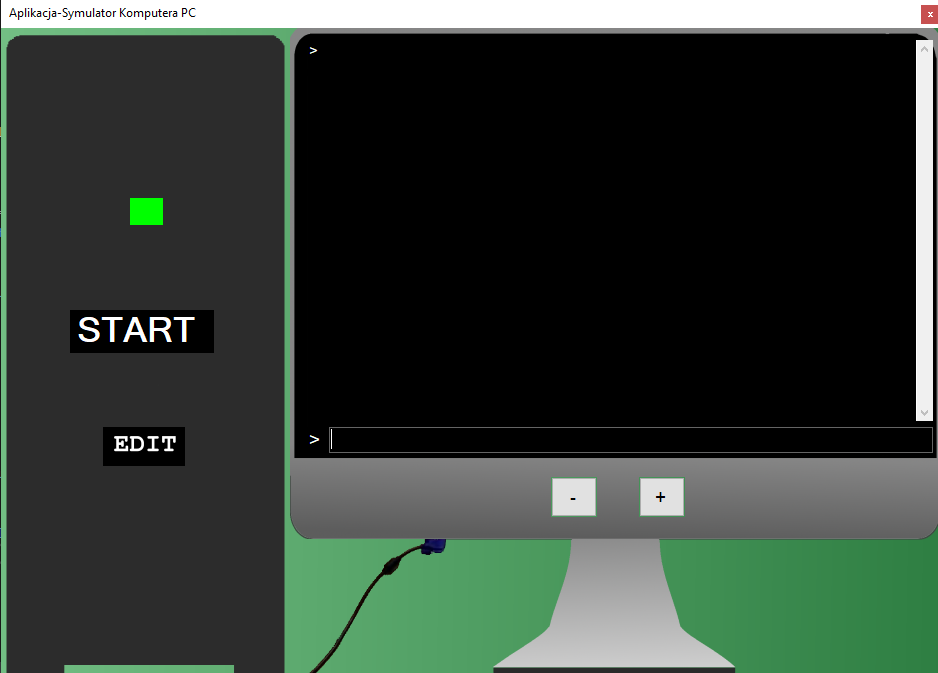
Po zakończonym włączaniu komputera, dioda zmieni kolor na zielony oraz funkcja StartButton\_Click sprawdzi czy monitor jest podłączony i jeśli warunek zostanie spełniony to wyświetli terminal. Następnie funkcja sprawdza czy klawiatura jest podłączona. Gdy klawiatura nie będzie podłączona, to mimo że konsola i miejsce do wpisywania komend będzie wyświetlone to użytkownik nie będzie mógł nic wpisać.



Rysunek 8: Fragment funkcji StartButton\_Click odpowiadający za proces wyświetlenia terminala komputera

Źródło: opracowanie własne

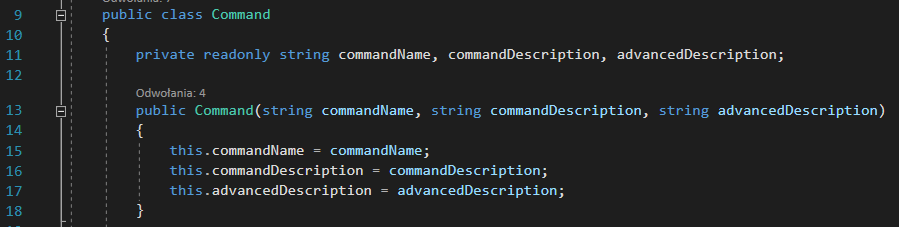
Jeśli wszystkie komponenty są podłączone oraz monitor i klawiatura, to po uruchomieniu komputera, wyświetli się ekran z terminalem.



Rysunek 8: Okno główne z włączonym komputerem

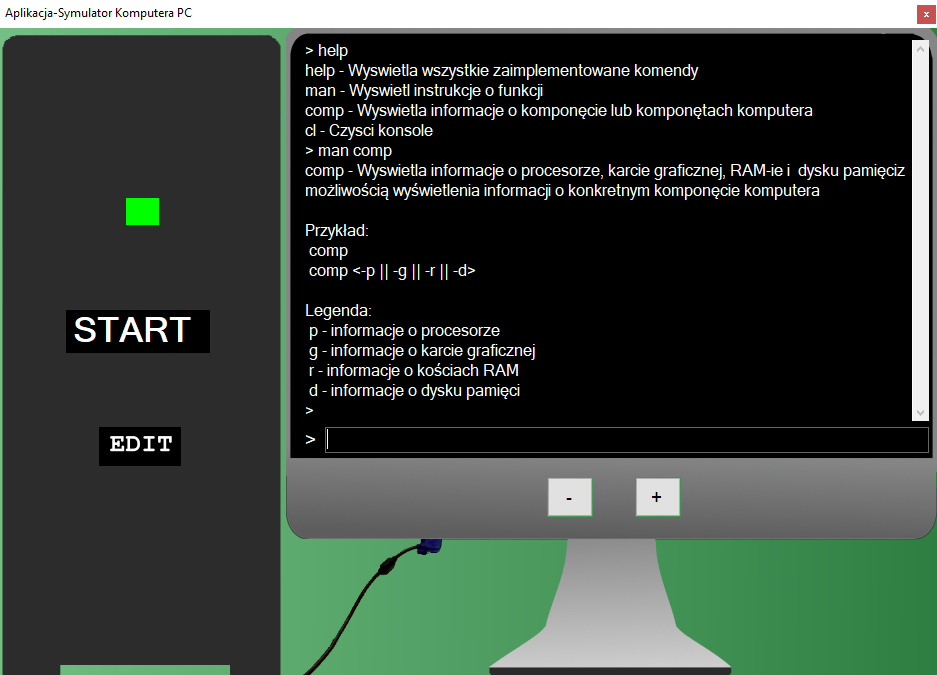
Źródło: opracowanie własne

Przyciski znajdujące się na dolnej części monitora odpowiadają za powiększenie (+) i pomniejszenie (-) czcionki terminala. Terminal posiada cztery komendy z których użytkownik może korzystać. Aby wypisać listę komend użytkownik musi wprowadzić komendę „help”. Każda komenda jest obiektem klasy Command która posiada trzy pola readonly typu string, odpowiadający kolejno za: nazwę komendy, krótki opis komendy wykorzystywany przy komendzie help oraz dłuższy opis komendy. Dłuższy opis komendy jest wykorzystywany przy użyciu komendy „man” która wypisuje instrukcje komendy.

Rysunek 9: Pola i konstruktor klasy Command

Źródło: opracowanie własne

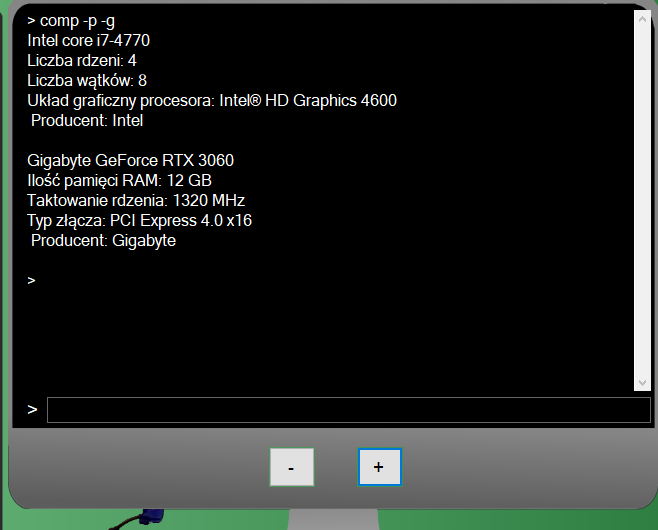
### 



Rysunek 10: Użycie komendy help oraz komendy man comp, która wypisuje instrukcje do komendy comp

Źródło: opracowanie własne

Komenda „cl” czyści konsole ze wszystkich wiadomości i tekstu. Najważniejszą komendą jest komenda „comp” która służy do wypisywania aktualnie wybranych komponentów symulowanego komputera. Jeśli zostanie wprowadzone tylko słowo „comp” to zostaną wypisane informacje o każdym z elementów komputera, które można wybierać. Jeśli do słowa „comp” dopiszemy odpowiednie prefix-y to otrzymamy informacje tylko i wyłącznie o elementach przypisanych do tych prefix-ów.



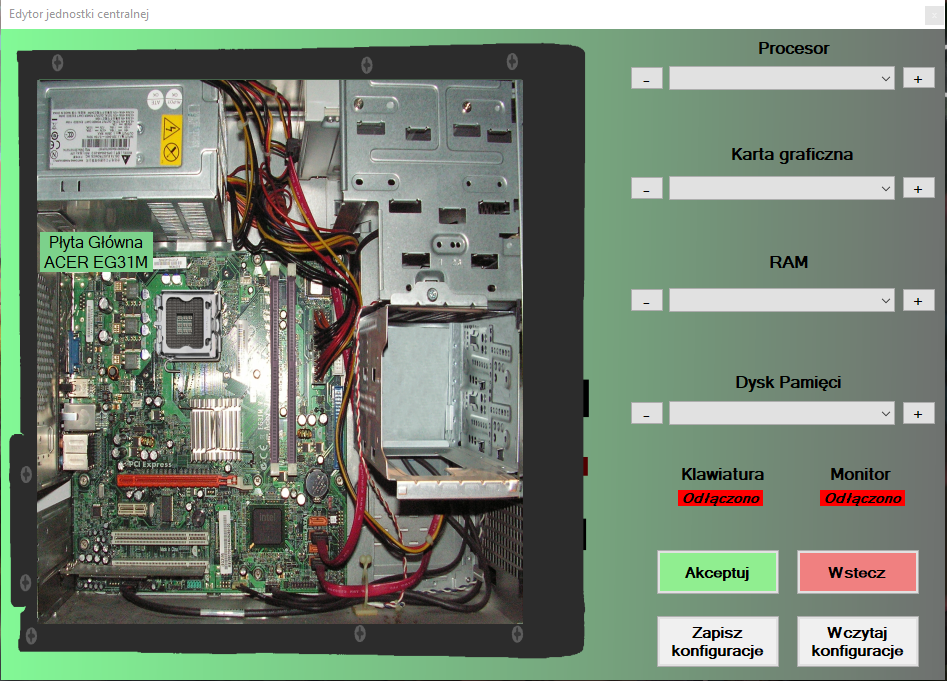
Rysunek 11. Użycie komendy comp z prefix-ami -p oraz -g

Źródło: opracowanie własne

Dzięki użyciu prefix-u użytkownik może wypisać tylko żądany element lub elementy, bez potrzeby wypisawania wszystkich komponentów podłączonych do symulowanego komputera. Komenda pokazuje w, że użytkownik ma wpływ na konfiguracje i działanie komputera. Jeśli dysk pamięci wybrany w konfiguracji jest dyskiem typu HDD czyli w jego nazwie występuje słowo HDD, to wtedy aplikacja będzie uruchamiała sie dłużej w porównaniu do dysku który będzie typu SSD i będzie posiadał w swojej nazwie słowo SSD.

### 3.2.2. Okno edycji komputera

Gdy komputer jest wyłączony, to po wciśnięciu przycisku „EDIT”, otworzy się okno edycji komputera w którym użytkownik może, zmieniać podłączone komponenty komputera.

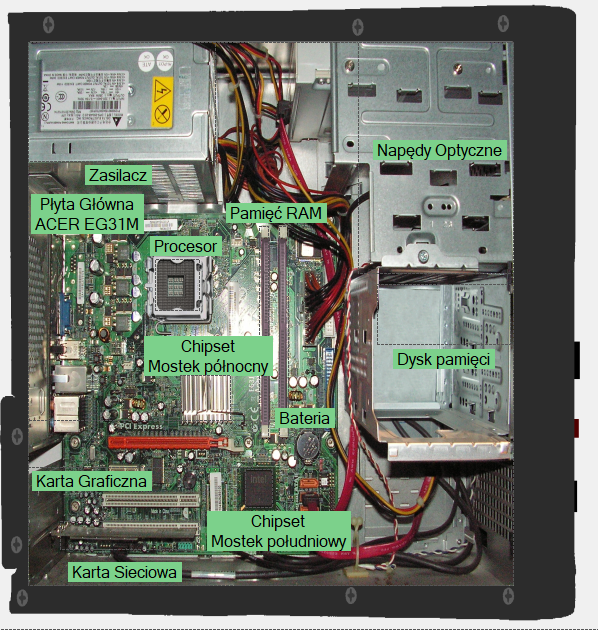


Rysunek 12. Okno edycji komputera

Źródło: opracowanie własne

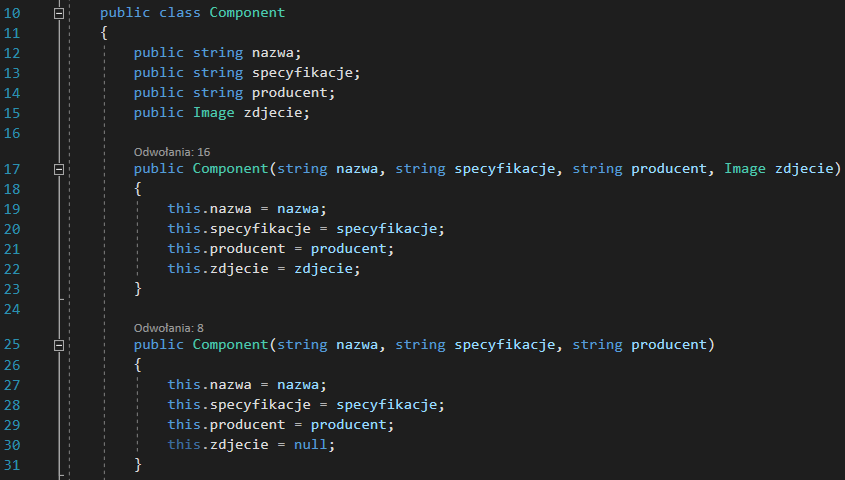
Jeśli w widoku komputera po lewej stronie najedziemy na któryś z ważniejszych elementów to wyświetli się nam nazwa tego komponentu.

Każdy komponent jest obiektem klasy Component która posiada pole odpowiadające za nazwę komponentu, jego specyfikacje, nazwę twórcy elementu oraz pole na zdjęcie, które zostanie użyte w poglądzie w trybie edycji. Jeśli komponent nie będzie posiadał zdjęcia, wtedy zostanie użyty drugi konstruktor który nie potrzebuje zdjęcia, aby utworzyć obiekt.



Rysunek 13. Podgląd programistyczny wnętrza

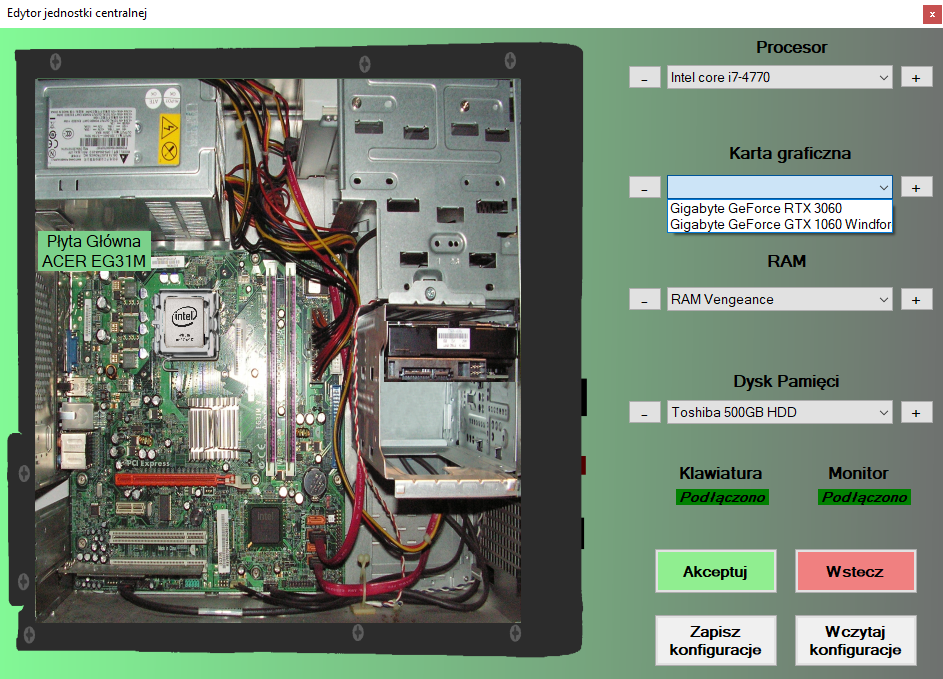
Źródło: opracowanie własne

Rysunek 14: Klasa Component

Źródło: opracowanie własne

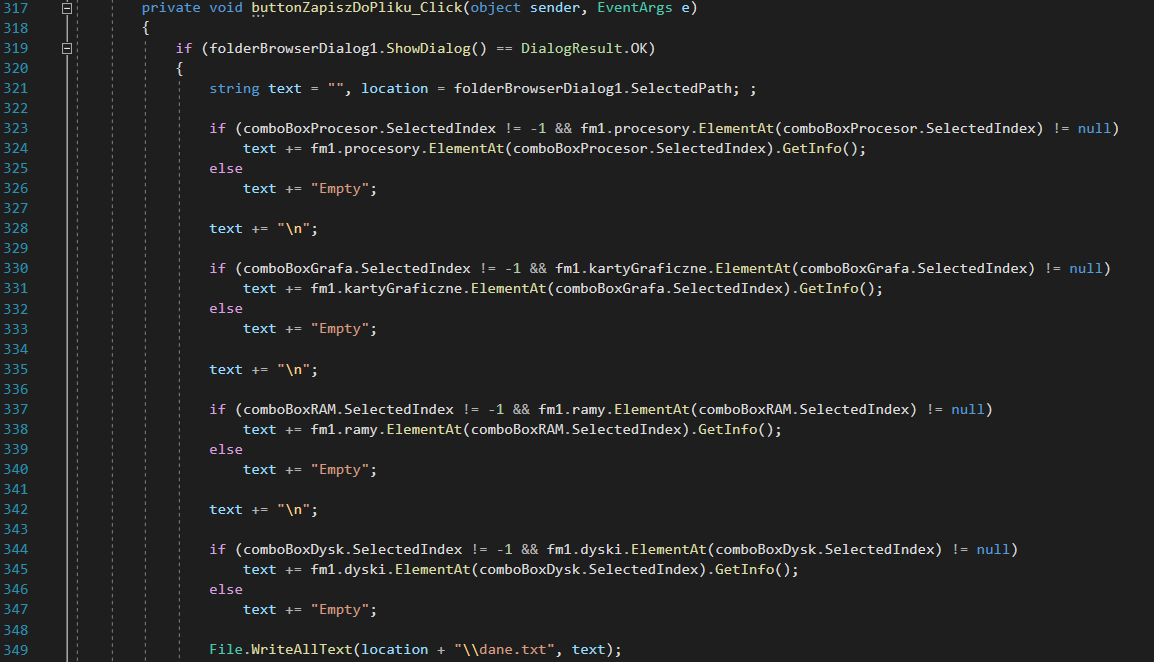
W oknie edycji użytkownik może wybrać komponenty z rozsuwanych list, po czym komponent graficznie jest umieszczany na podglądzie oraz jest przydzielany do odpowiedniej zmiennej globalnej odpowiadającej za aktualnie podłączony element. Po kliknięciu na napisy pod słowami „Klawiatura” oraz „Monitor”, podłącza się lub odłącza odpowiednio klawiaturę lub monitor.

Przycisk „Akceptuj” zapisuje dokonane zmiany i wyłącza okno edycji, a przycisk „Wstecz” nie zapisuje żadnych zmian i również wyłącza okno edycji. Przycisk „Zapisz konfiguracje” powoduje zapisanie aktualnej konfiguracji komponentów w postaci jednego pliku tekstowego w którym są zapisane dane tekstowe komponentów. Do tej samej lokalizacji zapisane są zdjęcia każdego komponentu który został wybrany. Jeśli któreś z pól na elementy komputera jest puste i zostanie zapisana taka konfiguracja to w pliku tekstowym takie pole jest zapisane jako słowo „Empty”. Przy zapisywaniu aplikacja zastępuje w danej lokalizacji najpierw usuwając pliki o nazwach, które mają mieć zapisane pliki konfiguracyjne, a następnie zapisuje do tej lokalizacji pliki konfiguracji.



Rysunek 15. Okno edycji komputera z wybranym procesorem, pamięcią RAM, i dyskiem pamięci

Źródło: opracowanie własne



Rysunek 16. Fragment metody odpowiadającej za zapisanie konfiguracji do pliku tekstowego

Źródło: opracowanie własne

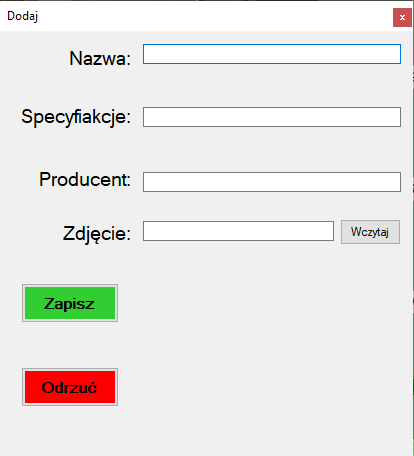
Przycisk „Wczytaj konfiguracje” również prosi użytkownika o wskazanie lokalizacji z której, ma być wczytana wcześniej zapisana konfiguracja. Zmiany w nazwach plików, rozszerzeniach lub w zawartości mogą skutkować nie wczytaniem pliku.

Obrazy będą zapisywane i wczytywane jako pliki z rozszerzeniem **.png**, ponieważ ten typ pliku może zostać zapisany wraz z przezroczystością kanału alpha.

Przycisk „-” służy do resetowania wybranego elementu z listy w taki sposób, żeby żaden komponent z danej listy nie był wybrany. Każdy taki przycisk odpowiada polu wyboru komponentu przy którymi się znajduje. obok siebie.

### 3.2.3. Okno dodania nowego komponentu

Po naciśnięciu przycisku „+” otworzy się okno dodawania nowego komponętu, dzięki któremu użytkownik może dodać własny komponent. Każdy przycisk „+” odpowiada kategorii elementu komputera przy której stoi.



Rysunek 17: Okno dodania nowego komponentu

Źródło: opracowanie własne

W rubrykach „Nazwa”, „Specyfikacje” i „Producent” użytkownik musi wpisać informacje o danym komponęcie, które będą wyświetlane jeśli dany komponęt zostanie wybrany w tej konfiguracji. W rubryce zdjęcie użytkownik może ręcznie wpisać lokalizacje pliku **.png** z dysku lub może użyć przycisku „Wczytaj” który otworzy okno wyboru pliku.

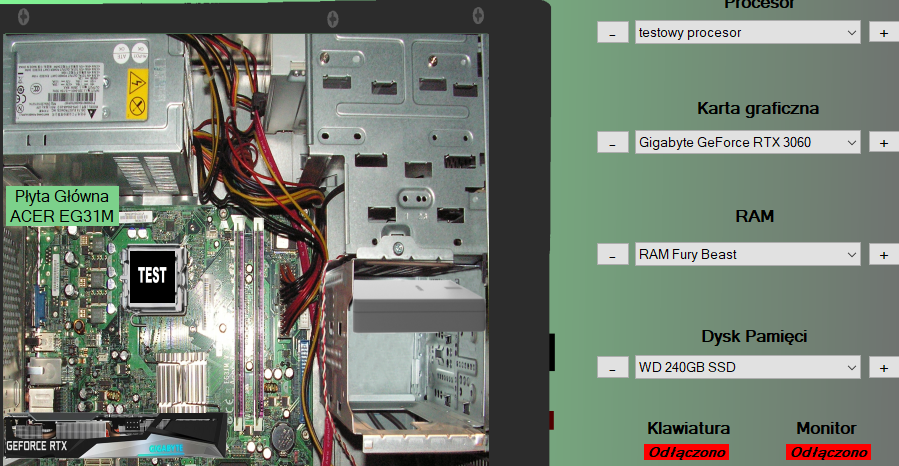
Po wybraniu pliku zdjęcie zostanie wyświetlone w pustym miejscu pod lokalizacją pliku.



Rysunek 18: Okno dodania komponentu z przykładowymi danymi

Źródło: opracowanie własne

Przycisk „Odrzuć” spowoduje zamknięcie okna i nie dodanie nowego elementu, za to przycisk „Zapisz”, zapisuje nowy element i dodaje go do danej listy adekwatnie do wybranego przycisku „+”



Rysunek 19: Konfiguracja z dodanym przykładowym komponentem

Źródło: opracowanie własne

# Podsumowanie

Celem pracy było wykonanie aplikacji symulującą zasadę działania komputera klasy PC w oparciu o jego układy blokowe wraz z wizualizacją poszczególnych elementów układu oraz przedstawienie organizacji i architektury klasycznego komputera klasy PC. Aplikacji ma za zdanie symulować działania klasycznego komputera osobistego z pokazaniem elementów budowy. Motywacją do wykonania omawianej pracy, była ciekawość budowy architektury i działania klasycznych komputerów osobistych.

Cel pracy został zrealizowany w pełni. Aplikacja pozwala na wybranie dowolnego elementu z każdej z czterech list odpowiadających rodzajowi komponentu komputera czyli procesorowi, karcie graficznej, pamięci RAM oraz dysku pamięci masowej. Po ustawieniu elementu w każdym z czterech rodzajów oraz „podłączeniu” monitora i klawiatury w aplikacji, użytkownik może „włączyć” komputer w oknie głównym po czym korzystać z dostępnych komend w terminalu komputera w celu między innymi wypisania informacji o komponentach znajdujących się w aktualnej konfiguracji komputera.

Aplikacja może być dalej rozwijana przez dodanie większej ilości rodzajów komponentów, dzięki czemu da to większą swobodę konfiguracji podczas wybierania poszczególnych elementów komputera. Konsole w symulowanym komputerze można poszerzyć o dodatkowe komendy, które przedstawią wpływ wybieranych elementów komputera na jego sposób działania i wydajność. Można również wprowadzić zapisywanie konfiguracji w chmurze lub bazie danych, aby użytkownik nie utracił swoich konfiguracji

.

# Bibliografia

1. Ryszard Tadeusiewicz, *Krótka historia informatyki,* Warszawa, RM, 2019,

ISBN 978-83-8151-084-4

1. James W. Cortada, *IBM: The Rise And Fall And Reinvention Of A Global Icon,* Cambridge, Massachusetts Institute of Technology, 2019, ISBN 9780262039444
2. William Aspray, *John Von Neumann And The Origins Of Modern Computing,* Cambridge, Massachusetts Institute of Technology, 1990, ISBN 0-262-01121-2
3. *Komputer osobisty*, <https://pl.wikipedia.org/wiki/Komputer_osobisty>, [11.01.2022]
4. *Altair 8800*, <https://pl.wikipedia.org/wiki/Altair_8800>, [11.01.2022]
5. Jarosław Gajek, *Wynalazki i proces twórczy Johna von Neumanna,* [http://kwasnicki.prawo.uni.wroc.pl/pliki/Gajek%20o%20John%20vo](http://kwasnicki.prawo.uni.wroc.pl/pliki/Gajek o John von)[%20Neumann.pdf](http://kwasnicki.prawo.uni.wroc.pl/pliki/Gajek o John von Neumann.pdf), [20.05.2022]
6. *Apple I*, <https://en.wikipedia.org/wiki/Apple_I>, [4.02.2022]
7. William Stallings, *Computer Organization And Architecture Designing For Performance Eighth Edition,* Upper Saddle River, Prentice Hall, 2010,

ISBN-13: 978-0-13-607373-4

1. William Stallings, *Computer Organization And Architecture Design For Performance Fourth Edition,* Upper Saddle River, Prentice Hall, 2010,

ISBN: 83-204-2517-4

1. Wojciech Głocki, *Układy cyfrowe podręcznik dla technikum*, Warszawa, Wydawnictwo Szkolne i Pedagogiczne, 1998, ISBN 83-02-06242-1
2. *Układy cyfrowe*, <https://pl.wikipedia.org/wiki/Układ_cyfrowy>, [24.06.2022]
3. *Układ sekwencyjny*, <https://pl.wikipedia.org/wiki/Układ_sekwencyjny>, [24.06.2022]
4. *Historia języka C#*, <https://docs.microsoft.com/pl-pl/dotnet/csharp/whats-new/csharp-> [version-history](https://docs.microsoft.com/pl-pl/dotnet/csharp/whats-new/csharp-version-history), [24.06.2022]
5. *C Sharp*, <https://pl.wikipedia.org/wiki/C_Sharp>, [24.06.2022]
6. *Omówienie platformy .NET Framework*, <https://docs.microsoft.com/pl-pl/dotnet/framework/get-started/overview>, [1.07.2022]
7. Andrew Troelesen, Philip Japikse, *Pro C# 7: With .NET and .NET Core*, Minneapolis, Apress, ISBN-13: 978-1-4842-3017-6
8. Mohammad Rahman, *C# Deconstructed,* New York, Apress, ISBN-13: 978-1-4302-6670-9
9. *.NET Framework*, <https://pl.wikipedia.org/wiki/.NET_Framework>, [1.07.2022]

# Spis rysunków

Rysunek 1: Altair 8800 z 8-calową stacją dyskietek, https://en.wikipedia.org/wiki/Altair\_8800, [11.01.2022]

Rysunek 2: IBM PC 5150 z monochromatycznym monitorem 5151, https://www.geekweb.pl/inne/kartka-z-kalendarza/item/412-39-lat-temu-[zaprezentowano-pierwszego-ibm-pc](https://www.geekweb.pl/inne/kartka-z-kalendarza/item/412-39-lat-temu-zaprezentowano-pierwszego-ibm-pc), [4.02.2022]

Rysunek 3: John von Neumann, https://pl.wikipedia.org/wiki/John\_von\_Neumann#/media/Plik:JohnvonNeumann-[LosAlamos.jpg](https://pl.wikipedia.org/wiki/John_von_Neumann" \l "/media/Plik:JohnvonNeumann-LosAlamos.jpg), [23.05.2022]

Rysunek 4: Struktura komputera ias, Computer Organization And Architecture Designing For Performance Eighth Edition

Rysunek 5: Okno główne aplikacji

Rysunek 6: Funkcja StartButton\_Click

Rysunek 7: Funkcja PutTaskDelay

Rysunek 8: Fragment funkcji StartButton\_Click odpowiadający za proces wyświetlenia terminala komputera

Rysunek 9: Pola i konstruktor klasy Command

Rysunek 10: Użycie komendy help oraz komendy man comp, która ma wypisać instrukcje do komendy comp

Rysunek 11: Użycie komendy comp z prefix-ami -p oraz -g

Rysunek 12: Okno edycji komputera

Rysunek 13: Podgląd programistyczny wnętrza komputera z okna edycji wraz z wszystkimi podpisami elementów

Rysunek 14: Klasa Component

Rysunek 15: Okno edycji komputera z wybranym procesorem, pamięcią RAM i dyskiem pamięci

Rysunek 16: Fragment metody odpowiadającej za zapisanie konfiguracji do pliku tekstowego

Rysunek 17: Okno dodania nowego komponentu

Rysunek 18: Okno dodania komponentu z przykładowymi danymi, opracowanie własne

Rysunek 19: Konfiguracja z dodanym przykładowym komponentem, opracowanie własne