Politechnika Wrocławska

Wydział Informatyki i Telekomunikacji

Architektura Komputerów 2

Projekt i implementacja procesora Z80

Autorzy:

Stanisław Strauchold 259142

Michał Tłuczek 259050

Prowadzący:

Dr hab. inż. Tadeusz Tomczak

Data:

07.06.2022

1. Streszczenie

W niniejszej pracy przedstawiona jest nasza wersja procesora Z80 wykonana z wykorzystaniem programu Logisim Evolution. W dokumencie omówione są autorskie implementacje poszczególnych elementów procesora, a także sposób działania zaprojektowanej przez nas jednostki. Ponadto praca zawiera wnioski i problemy napotkane przez grupę projektową podczas wykonywania zadania.

2. Spis treści

Lista sekcji:

Lista rysunków:

Lista tabel:

3. Wprowadzenie

Rezultatem naszej pracy jest zbudowany w programie Logisim Evolution procesor, którego architektura wzorowana jest architekturze oryginalnego Z80. Sterowanie jednostką odbywa się za pomocą kodów instrukcji (ang. *opcode*), które są zgodne z instrukcjami w procesorze Z80. Warto jednak zaznaczyć, że zaimplementowana został jedynie część instrukcji. Budowa naszej jednostki stara się odwzorować budowę oryginalnej jednostki. Z racji na ograniczenia z zakresu wiedzy na tym etapie nauki oraz czasu, implementowany przez nas procesor posiada jedynie część funkcjonalności, w które wyposażony jest procesor Z80. Aby dokładnie omówić efekty naszej pracy w kolejnych sekcjach znajduje się dokładny opis naszej implementacji poszczególnych elementów jednostki.

4. Implementacja

4.1. Architektura

W niniejszej sekcji przedstawione i opisane zostaną poszczególne elementy wchodzące w skład organizacji naszego układu. Są to: rejestry ogólnego oraz specjalnego przeznaczenia, rejestry instrukcji, jednostka sterująca, jednostka arytmetyczno logiczna(nazywana również ALU), pamięć ROM przechowująca instrukcje, pamięć RAM przechowująca dane oraz licznik programu.

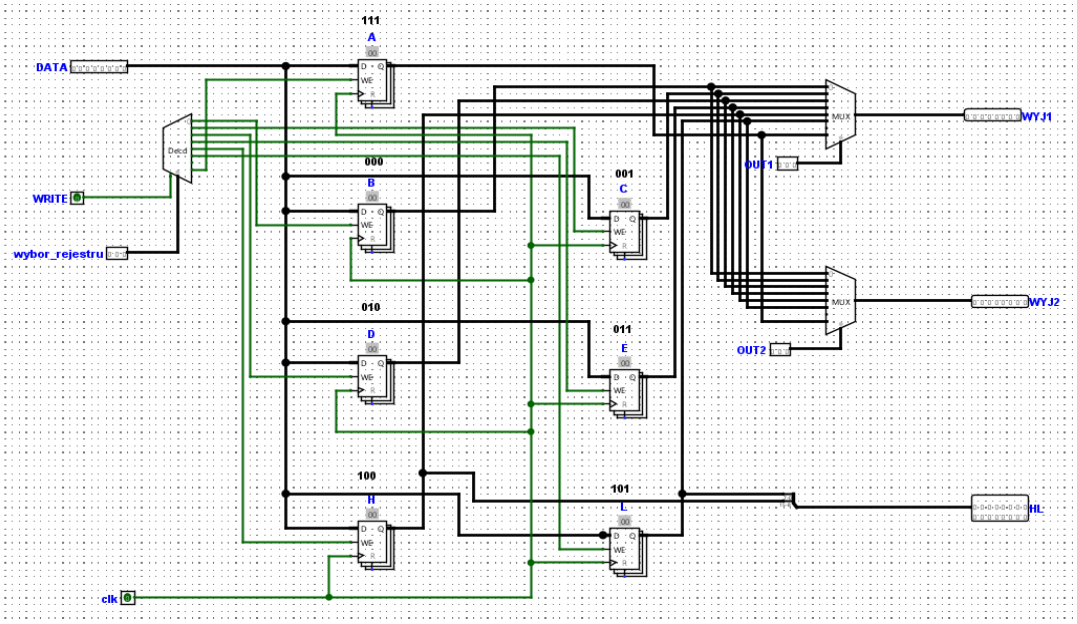
4.1.1. Rejestry

W skład zaimplementowanych przez nas rejestrów wchodzi sześć 8-bitowych rejestrów ogólnego przeznaczenia, z których dwa (H i L) mogą być połączone w parę i stworzyć rejestr 16-bitowy. Do tego jednostka wyposażona została w 8-bitowy akumulator oraz 6-bitowy rejestr flag. Procesor wyposażony jest również w licznik programu oraz rejestry instrukcji, których dokładny opis znajduje się w sekcji dotyczącej implementacji jednostki sterującej(patrz. *4.1.6. Jednostka sterująca*).

4.1.1.1. Rejestry ogólnego przeznaczenia

Oryginalny procesor Z80 wyposażony jest w dwa zestawy rejestrów ogólnego przeznaczenia. W naszej pracy zdecydowaliśmy się na umieszczenie tylko jednego z nich.

Rysunek poniżej przedstawia rejestry ogólnego przeznaczenia w wykonanym projekcie. Nazwy rejestrów są zgodne z nazwami stosowanymi w oryginalnej wersji implementowanego procesora. Na rysunku widoczny jest również akumulator(A) oraz rejestr HL.



Rysunek 1. Rejestry ogólnego przeznaczenia

4.1.1.2. Rejestry specjalnego przeznaczenia

W skład rejestrów specjalnego przeznaczenia w naszym procesorze wchodzą akumulator oraz rejestr flag. Oba te rejestry oryginalnie są 8-bitowe. W naszej implementacji zdecydowaliśmy si na pominięcie niewykorzystanych bitów w rejestrze flag, stąd też stworzony przez nas rejestr jest 6-bitowy. W rejestrze tym poszczególne bity odpowiedzialne są za odpowiednie flagi zgodnie z poniższym rysunkiem.

Rysunek 2. Rejestr flag

Poszczególne bity oznaczają następujące lagi:

S – flaga znaku

Z – flaga zero

H – flaga pół-przeniesienia

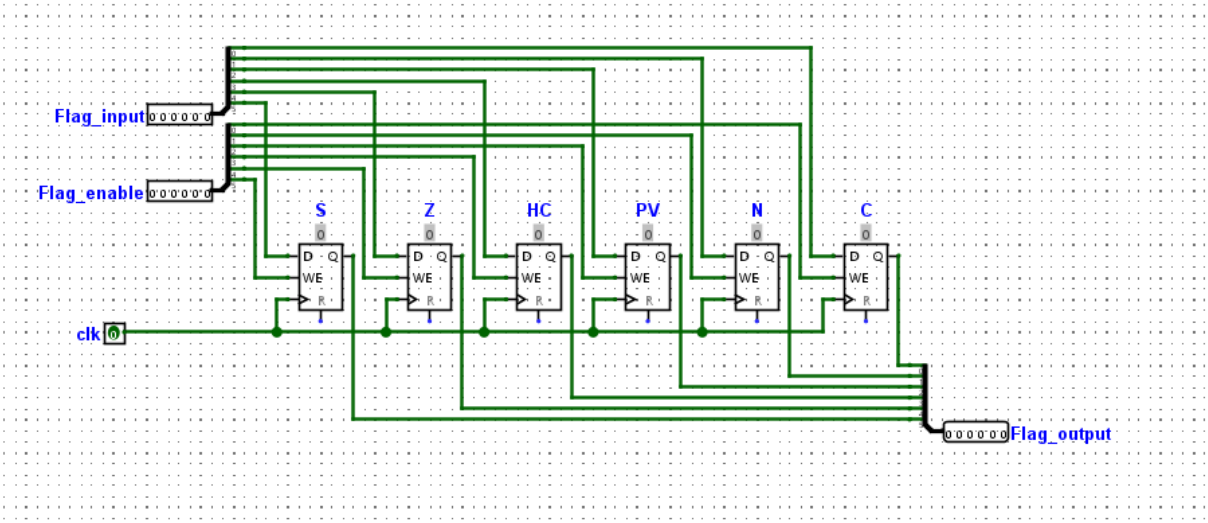
P – flaga parzystości/przepełnienia

N – flaga dodawanie/odejmowania

C – flaga przeniesienia

Bity oznaczone kreską nie mają określonego przeznaczenia i są przez nas pominięte

Rejestr flag w naszej implementacji prezentuje się następująco:



Rysunek 3. Nasz rejestr flag

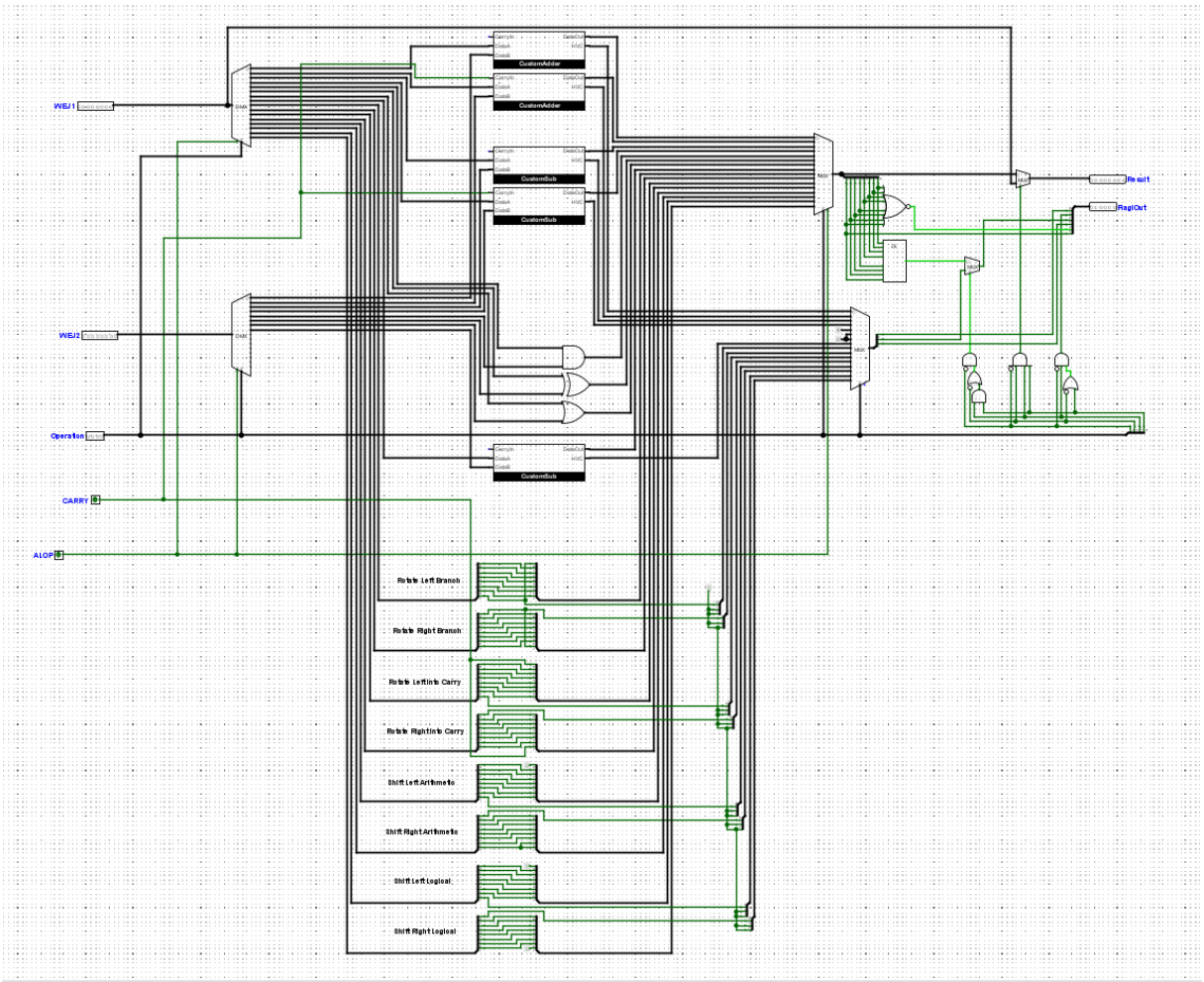
Akumulator widoczny był na Rysunku 1.

4.1.2. Jednostka arytmetyczno logiczna(ALU)

W ALU wykonywane są 8-bitowe operacje arytmetyczne i logiczne. Nasze ALU posiada zaimplementowane następujące operacje:

- Dodawanie
- Dodawanie z przeniesieniem
- Odejmowanie
- Odejmowanie z pożyczką
- Logiczny AND
- Logiczny OR
- Logiczny XOR
- Inkrementacja
- Dekrementacja
- Rotacja w prawo
- Rotacja w lewo
- Przesunięcie bitowe w prawo(arytmetyczne i logiczne)
- Przesunięcie bitowe w lewo(arytmetyczne i logiczne)

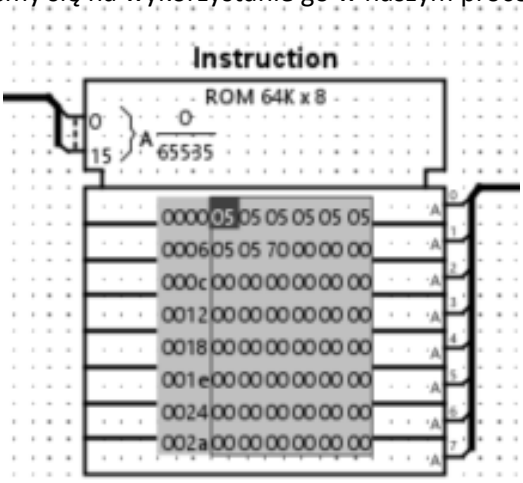
Warto tutaj nadmienić, że wyniki wszystkich operacji wykonywanych w ALU zapisywane są w akumulatorze



Rysunek 4 Nasza implementacja ALU

4.1.3. Pamięć ROM

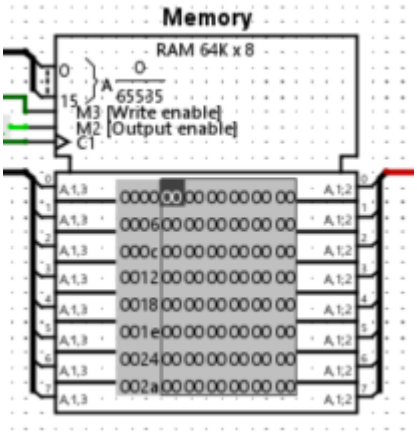
W pamięci ROM przechowywane są zapisane w reprezentacji szesnastkowej kody instrukcji. Z racji na występowanie w programie Logisim Evolution gotowego bloku reprezentującego pamięć ROM, zdecydowaliśmy się na wykorzystanie go w naszym procesorze.



Rysunek 5. Pamięć ROM

4.1.4. Pamięć RAM

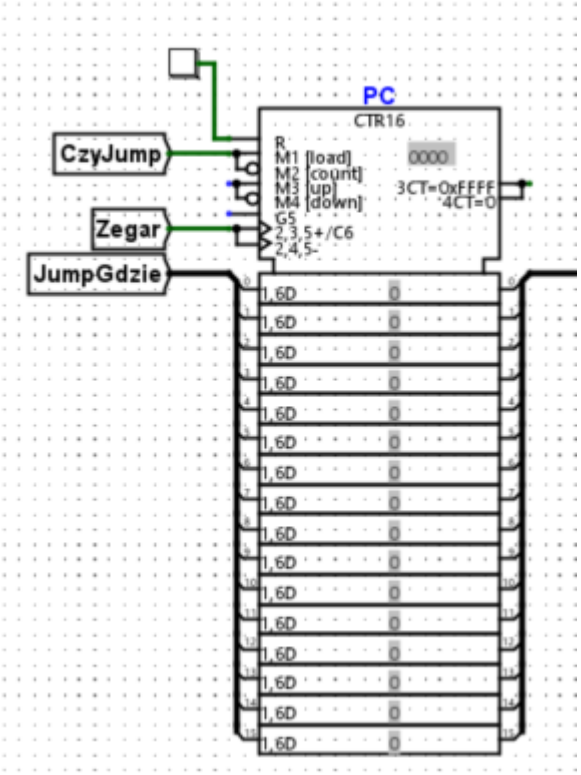
W pamięci RAM przechowywane są dane, które mogą być odczytywane i wykorzystywane przez procesor. Można również zapisywać w niej wyniki operacji arytmetycznych, czy zawartości poszczególnych rejestrów. Podobnie jak w przypadku pamięci ROM, do reprezentowania pamięci RAM posłużył gotowy blok dostępny w programie Logisim Evolution.



Rysunek 6. Pamięć RAM

4.1.5. Licznik rozkazów(PC)

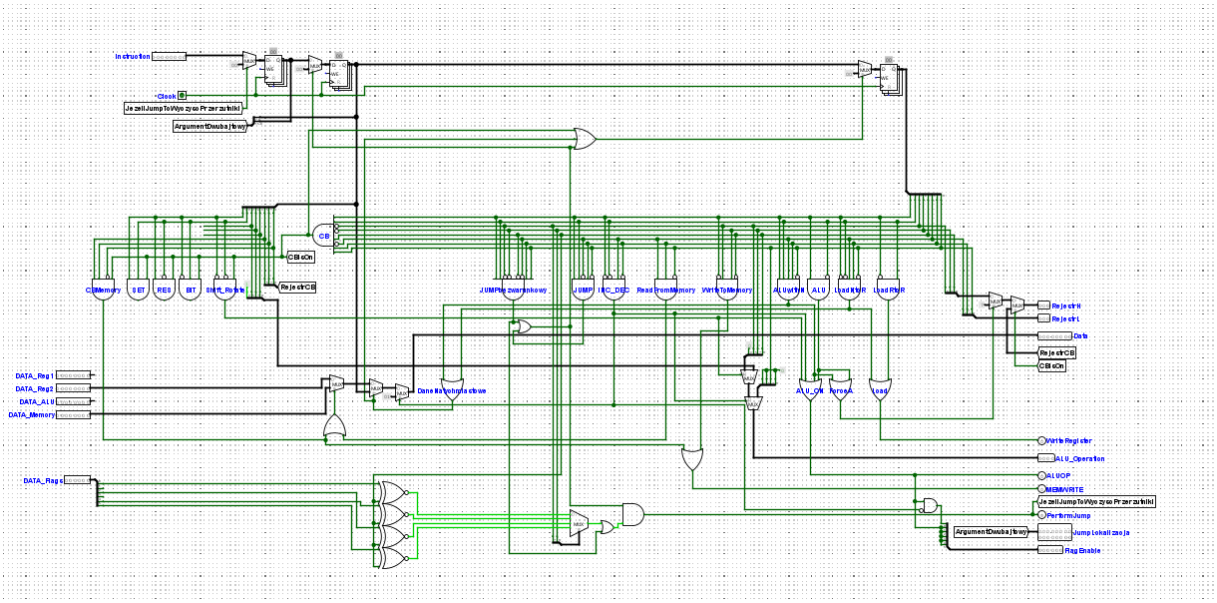
Licznik rozkazów przechowuje 16-bitowy adres aktualnie wykonywanej instrukcji. PC jest automatycznie inkrementowany za każdym razem, gdy jego zawartość zostanie przesłana na magistralę adresową. Podczas instrukcji skoku nowa wartość zostaje umieszczona w liczniku rozkazów, nadpisując przeniesienie. Do implementacji PC również posłużył gotowy blok dostępny w programie Logisim Evolution.



Rysunek 7. Licznik rozkazów(PC)

4.1.6. Jednostka sterująca

Jednostka sterująca odpowiedzialna jest za dekodowanie kodów instrukcji i sterowanie procesorem. W naszej implementacji w jej skład wchodzą trzy 8-bitowe rejestry, do których wczytywane są kolejne bajty instrukcji z pamięci ROM. Pozwala to na wykonywanie instrukcji, których kody są reprezentowane przez jeden, dwa oraz trzy bajty. Po wczytaniu tych wartości jednostka rozpoznaje kod i wykonuje odpowiednią operację.



Rysunek 8. Nasza jednostka sterująca

4.2. Tryby adresowania

W tej sekcji opisane zostaną tryby adresowania, które są zaimplementowane w naszym procesorze.

4.2.1. Natychmiastowy

W tym trybie adresowania bajt, który występuje w pamięci po kodzie instrukcji reprezentuje 8-bitowy argument, który jest wykorzystywany w instrukcji. Oznacza to, że instrukcje wykorzystujące 8-btowy argument natychmiastowy są 2 bajtowe.

4.2.2. Adresowanie rejestrowe

Adresowanie rejestrowe oznacza, że przy wykonywaniu instrukcji zostanie wykorzystany któryś z rejestrów. Kod instrukcji zawiera informację, który z rejestrów ma być użyty.

4.2.3. Adresowanie domniemane

W tym trybie adresowania kod instrukcji informuje, że operandem instrukcji będzie rejestr lub dwa rejestry.

4.2.4. Adresowanie pośrednie przez rejestr

Ten tryb adresowania wykorzystuje 16-bitową parę rejestrów , która przechowuje adres w pamięci, z którego dana ma być pobrana, lub do którego ma być zapisana. W naszej implementacji to adresowanie realizowane jest przez parę rejestrów HL.

4.3. Zestaw instrukcji

W tej sekcji opisane zostanie dokładne działanie zaimplementowanych przez nas instrukcji oraz opisany zostanie proces ich odczytywania z pamięci i dekodowania.

4.3.1. Ustawianie flag

Wiele z instrukcji zmienia zawartość rejestru flag. W celu lepszego zrozumienia ich działania na początku omówione zostaną flagi naszego procesora.

- Flaga Carry(C) – jest ustawiana lub czyszczona w zależności od rezultatu wykonywanej operacji. Flaga Carry jest ustawiana w przypadku generacji przeniesienia lub pożyczki w instrukcjach ADD i SUB. Analogicznie flaga ta jest resetowana w przypadku wygaszenia przeniesienia lub pożyczki. Flaga Carry jest także ustawiana w instrukcjach rotacji i przesunięć bitowych. Carry jest resetowana w przypadku instrukcji AND, OR i XOR
- Flaga N(dodawanie/odejmowanie) – flaga ta jest ustawiana po instrukcjach odejmowania i resetowana po instrukcjach dodawnia.
- Flaga P/V(parzystość/przepełnienie) – flaga ta jest ustawiana gdy wynik operacji arytmetycznej przekracza zakres akumulatora lub jest mniejszy niż najmniejszy możliwy numer zapisany w akumulatorze. Jest również ustawiana kiedy operacje logiczne i rotacje generują wynik parzysty lub nieparzysty. Dla wyniku parzystego P = 0, a dla nieparzystego P = 1.
- Flaga pół-przeniesienia(HC) – Flaga ta jest ustawiana lub resetowana w zależności od wytworzenia przeniesienia lub pożyczki pomiędzy 3 i 4 bitem operacji 8-bitowych.

- Flaga zero(Z) – jest ustawiana na 1 lub 0 w zależności od tego, czy wynikiem operacji jest 0, czy nie.
- Flaga znaku(S) – flaga ta informuje, czy wynik operacji arytmetycznej jest dodatni, czy ujemny. Dla wartości dodatnich flaga jest ustawiana na 0, a dla ujemnych na 1.

4.3.2. Grupy instrukcji

W tej sekcji znajdują się tabele prezentujące grupy instrukcji, które zostały przez nas zaimplementowane. Dla każdej instrukcji jest pokazany jej kod instrukcji w systemie szesnastkowym.

W niniejszych tabelach przyjęliśmy następujące oznaczenia:

A – akumulator

B, C, D, E, H, L – rejestry ogólnego przeznaczenia

(HL) – obszar w pamięci wskazywany przez parę rejestrów HL

n – 8-bitowy argument natychmiastowy w systemie szesnastkowym

nn – 16-bitowy argument natychmiastowy w systemie szesnastkowym

4.3.2.1. 8-bitowe instrukcje ładowania

Źródło

CEL	A	B	C	D	E	H	L	(HL)	n
A	7F	78	79	7A	7B	7C	7D	7E	3E n
B	47	40	41	42	43	44	45	46	06 n
C	4F	48	49	4A	4B	4C	4D	4E	0E n
D	57	50	51	52	53	54	55	56	16 n
E	5F	58	59	5A	5B	5C	5D	5E	1E n
H	67	60	61	62	63	64	65	66	26 n
L	6F	68	69	6A	6B	6C	6D	6E	2E n
(HL)	77	70	71	72	73	74	75		36 n

Tabela 1. 8-bitowe instrukcje ładowania danych

4.3.2.2. 8-bitowe instrukcje arytmetyczne i logiczne

Argument

Operacja	A	B	C	D	E	H	L	(HL)	n
ADD	87	80	81	82	83	84	85	86	C6 n
ADC	8F	88	89	8A	8B	8C	8D	8E	CE n
SUB	97	90	91	92	93	94	95	96	D6 n
SBC	9F	98	99	9A	9B	9C	9D	9E	DE n
AND	A7	A0	A1	A2	A3	A4	A5	A6	E6 n
XOR	AF	A8	A9	AA	AB	AC	AD	AE	EE n
OR	B7	B0	B1	B2	B3	B4	B5	B6	F6 n
INC	3C	4	0C	14	1C	24	2C	34	
DEC	3D	5	0D	15	1D	25	2D	35	

Tabela 2. 8-bitowe instrukcje arytmetyczne i logiczne

4.3.2.3. 8-bitowe instrukcje rotacji i przesunięć bitowych

Źródło

Operacja	A	B	C	D	E	H	L	(HL)
RLC	CB 07	CB 00	CB 01	CB 02	CB 03	CB 04	CB 05	CB 06
RRC	CB 0F	CB 08	CB 09	CB 0A	CB 0E	CB 0C	CB 0D	CB 0E
RL	CB 17	CB 10	CB 11	CB 12	CB 13	CB 14	CB 15	CB 16
RR	CB 1F	CB 18	CB 19	CB 1A	CB 1B	CB 1C	CB 1D	CB 1E
SLA	CB 27	CB 20	CB 21	CB 22	CB 23	CB 24	CB 25	CB 26
SRA	CB 2F	CB 28	CB 29	CB 2A	CB 2B	CB 2C	CB 2D	CB 2E
SRL	CB 3F	CB 38	CB 39	CB 3A	CB 3B	CB 3C	CB 3D	CB 3E

Tabela 3.8-bitowe Instrukcje rotacji i przesunięć bitowych

4.3.2.4. Instrukcje skoków

Skok	nn
JP nn	C3 nn
JP NZ nn	C2 nn
JP Z nn	CA nn
JP NC nn	D2 nn
JP C nn	DA nn
JP PO nn	E2 nn
JP PE nn	EA nn
JP P nn	F2 nn
JP M nn	FA nn

Tabela 4. Instrukcje skoków

4.3.2.5. Instrukcje porównania

Źródło

Operacja	A	B	C	D	E	H	L	(HL)	n
CP	BF	BB	B9	BA	BB	BC	BD	BE	FE n

Tabela 5. Instrukcje porównania

4.3.3. Opis zaimplementowanych instrukcji

Niniejsza sekcja zawiera opisy działania zaimplementowanych przez nas instrukcji

4.3.3.1. LD r, r’

Opis: ładuje zawartość rejestru r’ do rejestru r. Jako r traktujemy dowolny rejestr z grupy rejestrów ogólnego przeznaczenia oraz akumulator.

Kod instrukcji: 0 1 r r r r’ r’ r’

Identyfikacja rejestrów:

A – 111

B – 000 C – 001

D – 010 E – 011

H – 100 L – 101

Zmiana flag: nie dotyczy

4.3.3.2. LD r, n

Opis: ładuje argument natychmiastowy n do rejestru r.

Kod instrukcji: 0 0 r r r 1 1 0 n n n n n n n n

Identyfikacja rejestrów:

A – 111

B – 000 C – 001

D – 010 E – 011

H – 100 L – 101

Zmiana flag: nie dotyczy

4.3.3.3. LD r, (HL)

Opis: ładuje zawartość pamięci adresowaną przez HL do rejestru r

Kod instrukcji: 0 1 r r r 1 1 0

Identyfikacja rejestrów:

A – 111

B – 000 C – 001

D – 010 E – 011

H – 100 L – 101

Zmiana flag: nie dotyczy

4.3.3.4. LD (HL), r

Opis: ładuje zawartość rejestru r do miejsca w pamięci wskazywanym przez HL

Kod instrukcji: 0 1 1 1 0 r r r

Identyfikacja rejestrów:

A – 111

B – 000 C – 001

D – 010 E – 011

H – 100 L – 101

Zmiana flag: nie dotyczy

4.3.3.5. LD (HL), n

Opis: ładuje argument natychmiastowy n do miejsca w pamięci wskazywanego przez HL

Kod instrukcji: 0 0 1 1 0 1 1 0 n n n n n n n n

Zmiana flag: nie dotyczy

4.3.3.6. ADD A, r

Opis: dodaje zawartość rejestru r i akumulatora. Wynik jest zapisywany w akumulatorze.

Kod instrukcji: 1 0 0 0 0 r r r

Identyfikacja rejestrów:

A – 111

B – 000 C – 001

D – 010 E – 011

H – 100 L – 101

Zmiana flag:

S – ustawiana jeżeli wynik jest ujemny, reset w przeciwnym przypadku

Z – ustawiana jeżeli wynik jest zerem, reset w przeciwnym przypadku

H – ustawiana jeżeli występuje przeniesienie z bitu 3, reset w przeciwnym przypadku

P/V – ustawiana w przypadku wystąpienia przepełnienia, reset w przeciwnym przypadku

N – reset

C – ustawiona jeżeli występuje przeniesienie z bitu 7, reset w przeciwnym przypadku

4.3.3.7. ADD A, n

Opis: dodaje argument natychmiastowy n do zawartości akumulatora. Zawartość jest zapisywana w akumulatorze.

Kod instrukcji: 1 1 0 0 0 1 1 0 n n n n n n n n

Zmiana flag:

S – ustawiana jeżeli wynik jest ujemny, reset w przeciwnym przypadku

Z – ustawiana jeżeli wynik jest zerem, reset w przeciwnym przypadku

H – ustawiana jeżeli występuje przeniesienie z bitu 3, reset w przeciwnym przypadku

P/V – ustawiana w przypadku wystąpienia przepełnienia, reset w przeciwnym przypadku

N – reset

C – ustawiona jeżeli występuje przeniesienie z bitu 7, reset w przeciwnym przypadku

4.3.3.8. ADD A, (HL)

Opis: dodaje zawartość pamięci wskazywaną przez parę rejestrów HL do akumulatora. Wynik zapisywany jest w akumulatorze.

Kod instrukcji: 1 0 0 0 0 1 1 0

Zmiana flag:

S – ustawiana jeżeli wynik jest ujemny, resetowana w przeciwnym przypadku

Z – ustawiana jeżeli wynikiem jest 0, resetowana w przeciwnym przypadku

H – ustawiana jeżeli występuje przeniesienie z bitu 3, resetowana w przeciwnym przypadku

P/V – ustawiana w przypadku wystąpienia przepełnienia, resetowana w przeciwnym pzrypadku

N – resetowana

C – ustawiana jeżeli występuje przeniesienie z bitu 7, resetowana w przeciwnym przypadku

4.3.3.9. ADC A, r

Opis: dodaje zawartość rejestru r i zawartość flagi Carry do zawartości akumulatora. Zapisuje wynik operacji w akumulatorze.

Kod instrukcji: 1 0 0 0 1 r r r

Identyfikacja rejestrów:

A – 111

B – 000 C – 001

D – 010 E – 011

H – 100 L – 101

Zmiana flag:

S – ustawiana jeżeli wynik jest ujemny, reset w przeciwnym przypadku

Z – ustawiana jeżeli wynik jest zerem, reset w przeciwnym przypadku

H – ustawiana jeżeli występuje przeniesienie z bitu 3, reset w przeciwnym przypadku

P/V – ustawiana w przypadku wystąpienia przepełnienia, reset w przeciwnym przypadku

N – reset

C – ustawiona jeżeli występuje przeniesienie z bitu 7, reset w przeciwnym przypadku

4.3.3.10. ADC A, n

Opis: dodaje argument natychmiastowy n i zawartość flagi Carry do zawartości akumulatora. Zapisuje wynik w akumulatorze.

Kod instrukcji: 1 1 0 0 1 1 1 0 n n n n n n n n

Zmiana flag:

S – ustawiana jeżeli wynik jest ujemny, reset w przeciwnym przypadku

Z – ustawiana jeżeli wynik jest zerem, reset w przeciwnym przypadku

H – ustawiana jeżeli występuje przeniesienie z bitu 3, reset w przeciwnym przypadku

P/V – ustawiana w przypadku wystąpienia przepełnienia, reset w przeciwnym przypadku

N – reset

C – ustawiona jeżeli występuje przeniesienie z bitu 7, reset w przeciwnym przypadku

4.3.3.11. ADC A, (HL)

Opis: dodaje wartość w pamięci wskazywaną przez parę rejestrów HL i zawartość flagi Carry do zawartości akumulatora. Zapisuje wynik w akumulatorze.

Kod instrukcji: 1 0 0 0 1 1 1 0

Zmiana flag:

S – ustawiana jeżeli wynik jest ujemny, reset w przeciwnym przypadku

Z – ustawiana jeżeli wynik jest zerem, reset w przeciwnym przypadku

H – ustawiana jeżeli występuje przeniesienie z bitu 3, reset w przeciwnym przypadku

P/V – ustawiana w przypadku wystąpienia przepełnienia, reset w przeciwnym przypadku

N – reset

C – ustawiona jeżeli występuje przeniesienie z bitu 7, reset w przeciwnym przypadku

4.3.3.12. SUB r

Opis: odejmuje zawartość rejestru r od akumulatora. Zapisuje wynik w akumulatorze.

Kod instrukcji: 1 0 0 1 0 r r r

Identyfikacja rejestrów:

A – 111

B – 000 C – 001

D – 010 E – 011

H – 100 L – 101

Zmiana flag:

S – ustawiana jeżeli wynik jest ujemny, resetowana w przeciwnym przypadku

Z – ustawiana jeżeli wynikiem jest 0, resetowana w przeciwnym przypadku

H – ustawiana jeżeli występuje pożyczka z bitu 4, resetowana w przeciwnym przypadku

P/V – ustawiana jeżeli występuje przepełnienie, resetowana w przeciwnym przypadku

N – ustawiana

C – ustawiana jeżeli występuje pożyczka, resetowana w przeciwnym przypadku

4.3.3.13. SUB n

Opis: odejmuje argument natychmiastowy n od zawartości akumulatora. Zapisuje wynik w akumulatorze.

Kod instrukcji: 1 1 0 1 0 1 1 0 n n n n n n n n

Zmiana flag:

S – ustawiana jeżeli wynik jest ujemny, resetowana w przeciwnym przypadku

Z – ustawiana jeżeli wynikiem jest 0, resetowana w przeciwnym przypadku

H – ustawiana jeżeli występuje pożyczka z bitu 4, resetowana w przeciwnym przypadku

P/V – ustawiana jeżeli występuje przepełnienie, resetowana w przeciwnym przypadku

N – ustawiana

C – ustawiana jeżeli występuje pożyczka, resetowana w przeciwnym przypadku

4.3.3.14. SUB (HL)

Opis: odejmuje wartość w pamięci wskazywaną przez parę rejestrów HL od akumulatora. Zapisuje wynik w akumulatorze.

Kod instrukcji: 1 0 0 1 0 1 1 0

Zmiana flag:

S – ustawiana jeżeli wynik jest ujemny, resetowana w przeciwnym przypadku

Z – ustawiana jeżeli wynikiem jest 0, resetowana w przeciwnym przypadku

H – ustawiana jeżeli występuje pożyczka z bitu 4, resetowana w przeciwnym przypadku

P/V – ustawiana jeżeli występuje przepełnienie, resetowana w przeciwnym przypadku

N – ustawiana

C – ustawiana jeżeli występuje pożyczka, resetowana w przeciwnym przypadku

4.3.3.15. SBC A, r

Opis: odejmuje zawartość rejestru r oraz zawartość flagi C od zawartości akumulatora. Zapisuje wynik w akumulatorze.

Kod instrukcji: 1 0 0 1 1 r r r

Identyfikacja rejestrów:

A – 111

B – 000 C – 001

D – 010 E – 011

H – 100 L – 101

Zmiana flag:

S – ustawiana jeżeli wynik jest ujemny, resetowana w przeciwnym przypadku

Z – ustawiana jeżeli wynikiem jest 0, resetowana w przeciwnym przypadku

H – ustawiana jeżeli występuje pożyczka z bitu 4, resetowana w przeciwnym przypadku

P/V – ustawiana jeżeli występuje przepełnienie, resetowana w przeciwnym przypadku

N – ustawiana

C – ustawiana jeżeli występuje pożyczka, resetowana w przeciwnym przypadku

4.3.3.16. SBC A, n

Opis: odejmuje argument natychmiastowy n oraz zawartość flagi C od akumulatora. Zapisuje wynik w akumulatorze.

Kod instrukcji: 1 1 0 1 1 1 1 0 n n n n n n n n

Zmiana flag:

S – ustawiana jeżeli wynik jest ujemny, resetowana w przeciwnym przypadku

Z – ustawiana jeżeli wynikiem jest 0, resetowana w przeciwnym przypadku

H – ustawiana jeżeli występuje pożyczka z bitu 4, resetowana w przeciwnym przypadku

P/V – ustawiana jeżeli występuje przepełnienie, resetowana w przeciwnym przypadku

N – ustawiana

C – ustawiana jeżeli występuje pożyczka, resetowana w przeciwnym przypadku

4.3.3.17. SBC A, (HL)

Opis: odejmuje wartość w pamięci wskazywaną przez parę rejestrów HL oraz zawartość flagi przeniesienia od akumulatora. Zapisuje wynik w akumulatorze.

Kod instrukcji: 1 0 0 1 1 1 1 0

Zmiana flag:

S – ustawiana jeżeli wynik jest ujemny, resetowana w przeciwnym przypadku

Z – ustawiana jeżeli wynikiem jest 0, resetowana w przeciwnym przypadku

H – ustawiana jeżeli występuje pożyczka z bitu 4, resetowana w przeciwnym przypadku

P/V – ustawiana jeżeli występuje przepełnienie, resetowana w przeciwnym przypadku

N – ustawiana

C – ustawiana jeżeli występuje pożyczka, resetowana w przeciwnym przypadku

4.3.3.18. AND r

Opis: logiczne AND pomiędzy zawartością akumulatora, a zawartością rejestru r. Wynik zapisywany jest w akumulatorze.

Kod instrukcji: 1 0 1 0 0 r r r

Identyfikacja rejestrów:

A – 111

B – 000 C – 001

D – 010 E – 011

H – 100 L – 101

Zmiana flag:

S – ustawiana jeżeli wynik jest ujemny, resetowana w przeciwnym przypadku

Z – ustawiana jeżeli wynikiem jest zero, resetowana w przeciwnym przypadku

H – ustawiana

P/V – ustawiana jeżeli występuje parzysta liczba jedynek, resetowana w przeciwnym przypadku

N – resetowana

C – resetowana

4.3.3.19. AND n

Opis: logiczne AND pomiędzy argumentem natychmiastowym n, a zawartością akumulatora. Wynik zapisywany jest w akumulatorze.

Kod instrukcji: 1 1 1 0 0 1 1 0 n n n n n n n n

Zmiana flag:

S – ustawiana jeżeli wynik jest ujemny, resetowana w przeciwnym przypadku

Z – ustawiana jeżeli wynikiem jest zero, resetowana w przeciwnym przypadku

H – ustawiana

P/V – ustawiana jeżeli występuje parzysta liczba jedynek, resetowana w przeciwnym przypadku

N – resetowana

C – resetowana

4.3.3.20. AND (HL)

Opis: logiczny AND pomiędzy zawartością akumulatora, a wartością w pamięci wskazywaną przez parę rejestrów HL. Wynik zapisywany jest w akumulatorze.

Kod instrukcji: 1 0 1 0 0 1 1 0

Zmiana flag:

S – ustawiana jeżeli wynik jest ujemny, resetowana w przeciwnym przypadku

Z – ustawiana jeżeli wynikiem jest zero, resetowana w przeciwnym przypadku

H – ustawiana

P/V – ustawiana jeżeli występuje parzysta liczba jedynek, resetowana w przeciwnym przypadku

N – resetowana

C – resetowana

4.3.3.21. OR r

Opis: logiczny OR pomiędzy zawartością akumulatora, a zawartością rejestru r. Wynik zapisywany jest w akumulatorze.

Kod instrukcji: 1 0 1 1 0 r r r

Identyfikacja rejestrów:

A – 111

B – 000 C – 001

D – 010 E – 011

H – 100 L – 101

Zmiana flag:

S – ustawiana jeżeli wynik jest ujemny, resetowana w przeciwnym przypadku

Z – ustawiana jeżeli wynikiem jest zero, resetowana w przeciwnym przypadku

H – resetowana

P/V – ustawiana jeżeli występuje parzysta liczba jedynek, resetowana w przeciwnym przypadku

N – resetowana

C – resetowana

4.3.3.22. OR n

Opis: logiczny OR pomiędzy argumentem natychmiastowym n, a zawartością akumulatora. Wynik zapisywany jest w akumulatorze.

Kod instrukcji: 1 1 1 1 0 1 1 0 n n n n n n n n

Zmiana flag:

S – ustawiana jeżeli wynik jest ujemny, resetowana w przeciwnym przypadku

Z – ustawiana jeżeli wynikiem jest zero, resetowana w przeciwnym przypadku

H – resetowana

P/V – ustawiana jeżeli występuje parzysta liczba jedynek, resetowana w przeciwnym przypadku

N – resetowana

C – resetowana

4.3.3.23. OR (HL)

Opis: logiczny OR pomiędzy zawartością akumulatora, a wartością w pamięci wskazywaną przez parę rejestrów HL. Wynik zapisywany jest w akumulatorze.

Kod instrukcji: 1 0 1 1 0 1 1 0

Zmiana flag:

S – ustawiana jeżeli wynik jest ujemny, resetowana w przeciwnym przypadku

Z – ustawiana jeżeli wynikiem jest zero, resetowana w przeciwnym przypadku

H – resetowana

P/V – ustawiana jeżeli występuje parzysta liczba jedynek, resetowana w przeciwnym przypadku

N – resetowana

C – resetowana

4.3.3.24. XOR r

Opis: logiczny XOR pomiędzy zawartością akumulatora, a zawartością rejestru r. Wynik operacji jest zapisywany w akumulatorze.

Kod instrukcji: 1 0 1 1 0 r r r

Identyfikacja rejestrów:

A – 111

B – 000 C – 001

D – 010 E – 011

H – 100 L – 101

Zmiana flag:

S – ustawiana jeżeli wynik jest ujemny, resetowana w przeciwnym przypadku

Z – ustawiana jeżeli wynikiem jest zero, resetowana w przeciwnym przypadku

H – resetowana

P/V – ustawiana jeżeli występuje parzysta liczba jedynek, resetowana w przeciwnym przypadku

N- resetowana

C – resetowana

4.3.3.25. XOR n

Opis: logiczny XOR pomiędzy zawartością akumulatora, a argumentem natychmiastowym n. Wynik operacji zapisywany jest w akumulatorze.

Kod instrukcji: 1 1 1 1 0 1 1 0 n n n n n n n n

Zmiana flag:

S – ustawiana jeżeli wynik jest ujemny, resetowana w przeciwnym przypadku

Z – ustawiana jeżeli wynikiem jest zero, resetowana w przeciwnym przypadku

H – resetowana

P/V – ustawiana jeżeli występuje parzysta liczba jedynek, resetowana w przeciwnym przypadku

N- resetowana

C – resetowana

4.3.3.26. XOR (HL)

Opis: logiczny XOR pomiędzy zawartością akumulatora, a wartością w pamięci wskazywaną przez parę rejestrów HL. Wynik zapisywany jest w akumulatorze.

Kod instrukcji: 1 0 1 0 1 1 1 0

Zmiana flag:

S – ustawiana jeżeli wynik jest ujemny, resetowana w przeciwnym przypadku

Z – ustawiana jeżeli wynikiem jest zero, resetowana w przeciwnym przypadku

H – resetowana

P/V – ustawiana jeżeli występuje parzysta liczba jedynek, resetowana w przeciwnym przypadku

N- resetowana

C – resetowana

4.3.3.27. INC r

Opis: inkrementacja zawartości rejestru r. Wynik zapisywany jest w tym samym rejestrze, którego zawartość jest inkrementowana.

Kod instrukcji: 0 0 r r r 1 0 0

Identyfikacja rejestrów:

A – 111

B – 000 C – 001

D – 010 E – 011

H – 100 L – 101

Zmiana flag:

S – ustawiana jeżeli wynik jest negatywny, resetowana w przeciwnym przypadku

Z – ustawiana jeżeli wynikiem jest 0, resetowana w przeciwnym przypadku

H – ustawiana jeżeli występuje przeniesienie z bitu 3, resetowana w przeciwnym przypadku

P/V – ustawiana w przypadku wystąpienia przepełnienia, reset w przeciwnym przypadku

N – resetowana

C – nie dotyczy

4.3.3.28. INC (HL)

Opis: inkrementuje wartość w pamięci wskazywaną przez parę rejestrów HL. Wynik zapisywany jest w tym samym miejscu w pamięci.

Kod instrukcji: 0 0 1 1 0 1 0 0

Zmiana flag:

S – ustawiana jeżeli wynik jest negatywny, resetowana w przeciwnym przypadku

Z – ustawiana jeżeli wynikiem jest 0, resetowana w przeciwnym przypadku

H – ustawiana jeżeli występuje przeniesienie z bitu 3, resetowana w przeciwnym przypadku

P/V – ustawiana w przypadku wystąpienia przepełnienia, reset w przeciwnym przypadku

N – resetowana

C – nie dotyczy

4.3.3.29. DEC r

Opis: dekrementacja zawartości rejestru r. Wynik operacji zapisywany jest w rejestrze, którego zawartość jest inkrementowana.

Kod instrukcji: 0 0 r r r 1 0 1

Identyfikacja rejestrów:

A – 111

B – 000 C – 001

D – 010 E – 011

H – 100 L – 101

Zmiana flag:

S – ustawiana jeżeli wynik jest negatywny, resetowana w przeciwnym przypadku

Z – ustawiana jeżeli wynikiem jest 0, resetowana w przeciwnym przypadku

H – ustawiana jeżeli występuje pożyczka z bitu 4, resetowana w przeciwnym przypadku

P/V – ustawiana w przypadku wystąpienia przepełnienia, reset w przeciwnym przypadku

N – ustawiana

C – nie dotyczy

4.3.3.30. DEC (HL)

Opis: dekrementuje wartość w pamięci wskazywaną przez parę rejestrów HL. Wynik zapisywany jest w tej samej lokalizacji w pamięci.

Kod instrukcji: 0 0 1 1 0 1 0 1

Zmiana flag:

S – ustawiana jeżeli wynik jest negatywny, resetowana w przeciwnym przypadku

Z – ustawiana jeżeli wynikiem jest 0, resetowana w przeciwnym przypadku

H – ustawiana jeżeli występuje przeniesienie z bitu 3, resetowana w przeciwnym przypadku

P/V – ustawiana w przypadku wystąpienia przepełnienia, reset w przeciwnym przypadku

N – resetowana

C – nie dotyczy

4.3.3.31. RLC r

Opis: zawartość akumulatora r jest rotowana w lewo o 1 bit. Zawartość bitu 7 jest kopiowana do flagi przeniesienia(carry), a także do bitu 0.

Kod instrukcji: 1 1 0 0 1 0 1 1 0 0 0 0 0 r r r

Identyfikacja rejestrów:

A – 111

B – 000 C – 001

D – 010 E – 011

H – 100 L – 101

Zmiana flag:

S – ustawiana jeżeli wynik jest ujemny, resetowana w przeciwnym przypadku

Z – ustawiana jeżeli wynikiem jest zero, resetowana w przeciwnym przypadku

H – resetowana

P/V – ustawiana jeżeli występuje parzysta liczba jedynek, resetowana w przeciwnym przypadku

N – resetowana

C – jest zawartością bitu 7 rejestru źródłowego

4.3.3.32. RL r

Opis: zawartość rejestru r jest rotowana w lewo o 1 bit. Zawartość bitu 7 jest kopiowana do flagi przeniesienia(carry), a poprzednia zawartość flagi przeniesienia jest ładowana do bitu 0.

Kod instrukcji: 1 1 0 0 1 0 1 1 0 0 0 1 0 r r r

Identyfikacja rejestrów:

A – 111

B – 000 C – 001

D – 010 E – 011

H – 100 L – 101

Zmiana flag:

S – ustawiana jeżeli wynik jest ujemny, resetowana w przeciwnym przypadku

Z – ustawiana jeżeli wynikiem jest 0, resetowana w przeciwnym przypadku

H – resetowana

P/V – ustawiana jeżeli występuje parzysta liczba jedynek, resetowana w przeciwnym przypadku

N – resetowana

C – zawartość 7 bitu rejestru źródłowego

4.3.3.33. RRC r

Opis: zawartość rejestru r jest rotowana w prawo o 1 bit. Zawartość bitu 0 jest kopiowana do flagi przeniesienia(carry) oraz do bitu 7.

Kod instrukcji: 1 1 0 0 1 0 1 1 0 0 0 0 1 r r r

Identyfikacja rejestrów:

A – 111

B – 000 C – 001

D – 010 E – 011

H – 100 L – 101

Zmiana flag:

S – ustawiana jeżeli wynik jest ujemny, resetowana w przeciwnym przypadku

Z – ustawiana jeżeli wynikiem jest 0, resetowana w przeciwnym przypadku

H – resetowana

P/V – ustawiana jeżeli występuje parzysta liczba jedynek, resetowana w przeciwnym przypadku

N – resetowana

C – zawartość bitu 0 rejestru źródłowego

4.3.3.34. RR r

Opis: zawartość rejestru r jest rotowana w prawo o 1 bit przez flagę przeniesienia. Zawartość bitu 0 jest kopiowana do flagi przeniesienia, a jej poprzednia zawartość jest kopiowana do bitu 7.

Kod instrukcji: 1 1 0 0 1 0 1 1 0 0 0 0 1 r r r

Identyfikacja rejestrów:

A – 111

B – 000 C – 001

D – 010 E – 011

H – 100 L – 101

Zmiana flag:

S – ustawiana jeżeli wynik jest ujemny, resetowana w przeciwnym przypadku

Z – ustawiana jeżeli wynikiem jest 0, resetowana w przeciwnym przypadku

H – resetowana

P/V – ustawiana jeżeli występuje parzysta liczba jedynek, resetowana w przeciwnym przypadku

N – resetowana

C- zawartość bitu 0 rejestru źródłowego

4.3.3.35. SLA r

Opis: arytmetyczne lewe przesunięcie o 1 bit zawartości rejestru r. Zawartość bitu 7 jest kopiowana do flagi przeniesienia. Bit zero jest resetowany.

Kod instrukcji: 1 1 0 0 1 0 1 1 0 0 1 0 0 r r r

Identyfikacja rejestrów:

A – 111

B – 000 C – 001

D – 010 E – 011

H – 100 L – 101

Zmiana flag:

S – ustawiana jeżeli wynik jest ujemny, resetowana w przeciwnym przypadku

Z – ustawiana jeżeli wynikiem jest 0, resetowana w przeciwnym przypadku

H – resetowana

P/V – ustawiana jeżeli występuje parzysta liczba jedynek, resetowana w przeciwnym przypadku

N – resetowana

C- zawartość bitu 7 rejestru źródłowego

4.3.3.36. SRA r

Opis: przesunięcie arytmetyczne w prawo o 1 bit. Zawartość bitu 0 jest kopiowana do flagi przeniesienia. Poprzednia zawartość bitu 7 pozostaje niezmieniona.

Kod instrukcji: 1 1 0 0 1 0 1 1 0 0 1 0 1 r r r

Identyfikacja rejestrów:

A – 111

B – 000 C – 001

D – 010 E – 011

H – 100 L – 101

Zmiana flag:

S – ustawiana jeżeli wynik jest ujemny, resetowana w przeciwnym przypadku

Z – ustawiana jeżeli wynikiem jest 0, resetowana w przeciwnym przypadku

H – resetowana

P/V – ustawiana jeżeli występuje parzysta liczba jedynek, resetowana w przeciwnym przypadku

N – resetowana

C- zawartość bitu 0 rejestru źródłowego

4.3.3.37. SRL r

Opis: zawartość rejestru r jest przesuwana w prawo o 1 bit. Zawartość bitu 0 jest kopiowana do flagi przeniesienia, a bit 7 jest resetowany.

Kod instrukcji: 1 1 0 0 1 0 1 1 0 0 1 1 1 r r r

Identyfikacja rejestrów:

A – 111

B – 000 C – 001

D – 010 E – 011

H – 100 L – 101

Zmiana flag:

S – resetowana

Z – ustawiana jeżeli wynikiem jest 0, resetowana w przeciwnym przypadku

H – resetowana

P/V – ustawiana jeżeli występuje parzysta liczba jedynek, resetowana w przeciwnym przypadku

N – jest resetowana

C – dane z bitu 0 rejestru źródłowego

4.3.3.38. JP nn

Opis: 16-bitowy argument nn jest ładowany do licznika programu(PC). Następna instrukcja jest pobierana z lokalizacji w pamięci wskazywanej przez nową zawartość PC. Przy podawaniu adresu instrukcji należy najpierw podać mniej znaczący bajt adresu.

Kod instrukcji: 1 1 0 0 0 0 1 1 n n n n n n n n n n n n n n n n

Zmiana flag: nie dotyczy

4.3.3.39. JP cc, nn

Opis: jeżeli warunek cc jest spełniony, 16-bitowy argument nn jest ładowany do licznika programu(PC). Jeżeli warunek cc nie jest spełniony, argument jest ignorowany. Przy podawaniu adresu instrukcji należy najpierw podać mniej znaczący bajt adresu.

Kod instrukcji: 1 1 c c c 0 1 0 n n n n n n n n n n n n n n n n

Identyfikacja warunków:

000 – nie zero(NZ)

001 – zero(Z)

010 – brak przeniesienia(NC)

011 – przeniesienie(C)

100 – nieparzysty(PO)

101 – parzysty(PE)

110 – znak dodatni(P)

111 – znak ujemny(M)

Zmiana flag: nie dotyczy

4.3.3.40. CP r

Opis: odjęcie zawartości rejestru r od zawartości akumulatora i ustawienie flag w ten sposób aby określały, czy oba argumenty były równe. Wynik nie jest nigdzie zapisywany.

Kod instrukcji: 1 0 1 1 1 r r r

Identyfikacja rejestrów:

A – 111

B – 000 C – 001

D – 010 E – 011

H – 100 L – 101

Zmiana flag:

S – ustawiana jeżeli wynik jest ujemny, resetowana w przeciwnym przypadku

Z – ustawiana jeżeli wynikiem jest 0, resetowana w przeciwnym przypadku

H – ustawiana jeżeli występuje pożyczka z bitu 4, resetowana w przeciwnym przypadku

P/V – jest ustawiania w przypadku przepełnienia, resetowana w przeciwnym przypadku

N – ustawiana

C – ustawiana jeżeli występuje pożyczka, resetowana w przeciwnym przypadku

4.3.3.41. CP n

Opis: Opis: odjęcie zawartości argumentu natychmiastowego n od zawartości akumulatora i ustawienie flag w ten sposób aby określały, czy oba argumenty były równe. Wynik nie jest nigdzie zapisywany.

Kod instrukcji: 1 1 1 1 1 1 1 0 n n n n n n n n

Zmiana flag:

S – ustawiana jeżeli wynik jest ujemny, resetowana w przeciwnym przypadku

Z – ustawiana jeżeli wynikiem jest 0, resetowana w przeciwnym przypadku

H – ustawiana jeżeli występuje pożyczka z bitu 4, resetowana w przeciwnym przypadku

P/V – jest ustawiania w przypadku przepełnienia, resetowana w przeciwnym przypadku

N – ustawiana

C – ustawiana jeżeli występuje pożyczka, resetowana w przeciwnym przypadku

4.3.3.42. CP (HL)

Opis: odjęcie wartości w pamięci wskazywanej przez parę rejestrów HL od zawartości akumulatora i ustawienie flag w ten sposób aby określały, czy oba argumenty były równe. Wynik nie jest nigdzie zapisywany.

Kod instrukcji: 1 0 1 1 1 1 1 0

Zmiana flag:

S – ustawiana jeżeli wynik jest ujemny, resetowana w przeciwnym przypadku

Z – ustawiana jeżeli wynikiem jest 0, resetowana w przeciwnym przypadku

H – ustawiana jeżeli występuje pożyczka z bitu 4, resetowana w przeciwnym przypadku

P/V – jest ustawiania w przypadku przepełnienia, resetowana w przeciwnym przypadku

N – ustawiana

C – ustawiana jeżeli występuje pożyczka, resetowana w przeciwnym przypadku

4.3.3.43. RLC (HL)

Opis: wartość w pamięci wskazywana przez parę rejestrów HL jest rotowana w lewo o 1 bit. Zawartość bitu 7 jest kopiowana do flagi przeniesienia(carry), a także do bitu 0.

Kod instrukcji: 1 1 0 0 1 0 1 1 0 0 0 0 0 1 1 0

Identyfikacja rejestrów:

A – 111

B – 000 C – 001

D – 010 E – 011

H – 100 L – 101

Zmiana flag:

S – ustawiana jeżeli wynik jest ujemny, resetowana w przeciwnym przypadku

Z – ustawiana jeżeli wynikiem jest zero, resetowana w przeciwnym przypadku

H – resetowana

P/V – ustawiana jeżeli występuje parzysta liczba jedynek, resetowana w przeciwnym przypadku

N – resetowana

C – jest zawartością bitu 7 rejestru źródłowego

4.3.3.44. RRC (HL)

Opis: wartość w pamięci adresowana przez parę rejestrów HL jest rotowana w prawo o 1 bit. Zawartość bitu 0 jest kopiowana do flagi przeniesienia(carry) oraz do bitu 7.

Kod instrukcji: 1 1 0 0 1 0 1 1 0 0 0 0 1 1 1 0

Identyfikacja rejestrów:

A – 111

B – 000 C – 001

D – 010 E – 011

H – 100 L – 101

Zmiana flag:

S – ustawiana jeżeli wynik jest ujemny, resetowana w przeciwnym przypadku

Z – ustawiana jeżeli wynikiem jest 0, resetowana w przeciwnym przypadku

H – resetowana

P/V – ustawiana jeżeli występuje parzysta liczba jedynek, resetowana w przeciwnym przypadku

N – resetowana

4.3.3.45. RL (HL)

Opis: wartość w pamięci wskazywana przez parę rejestrów HL jest rotowana w lewo o 1 bit. Zawartość bitu 7 jest kopiowana do flagi przeniesienia(carry), a poprzednia zawartość flagi przeniesienia jest ładowana do bitu 0.

Kod instrukcji: 1 1 0 0 1 0 1 1 0 0 0 1 0 1 1 0

Identyfikacja rejestrów:

A – 111

B – 000 C – 001

D – 010 E – 011

H – 100 L – 101

Zmiana flag:

S – ustawiana jeżeli wynik jest ujemny, resetowana w przeciwnym przypadku

Z – ustawiana jeżeli wynikiem jest 0, resetowana w przeciwnym przypadku

H – resetowana

P/V – ustawiana jeżeli występuje parzysta liczba jedynek, resetowana w przeciwnym przypadku

N – resetowana

C – zawartość 7 bitu rejestru źródłowego

4.3.3.46. RR (HL)

Opis: wartość w pamięci wskazywana przez parę rejestrów HL jest rotowana w prawo o 1 bit przez flagę przeniesienia. Zawartość bitu 0 jest kopiowana do flagi przeniesienia, a jej poprzednia zawartość jest kopiowana do bitu 7.

Kod instrukcji: 1 1 0 0 1 0 1 1 0 0 0 1 1 1 1 0

Identyfikacja rejestrów:

A – 111

B – 000 C – 001

D – 010 E – 011

H – 100 L – 101

Zmiana flag:

S – ustawiana jeżeli wynik jest ujemny, resetowana w przeciwnym przypadku

Z – ustawiana jeżeli wynikiem jest 0, resetowana w przeciwnym przypadku

H – resetowana

P/V – ustawiana jeżeli występuje parzysta liczba jedynek, resetowana w przeciwnym przypadku

N – resetowana

C- zawartość bitu 0 rejestru źródłowego

4.3.3.47. SLA (HL)

Opis: arytmetyczne lewe przesunięcie o 1 bit wartości w pamięci adresowanej przez parę rejestrów HL. Zawartość bitu 7 jest kopiowana do flagi przeniesienia. Bit zero jest resetowany.

Kod instrukcji: 1 1 0 0 1 0 1 1 0 0 1 0 0 1 1 0

Identyfikacja rejestrów:

A – 111

B – 000 C – 001

D – 010 E – 011

H – 100 L – 101

Zmiana flag:

S – ustawiana jeżeli wynik jest ujemny, resetowana w przeciwnym przypadku

Z – ustawiana jeżeli wynikiem jest 0, resetowana w przeciwnym przypadku

H – resetowana

P/V – ustawiana jeżeli występuje parzysta liczba jedynek, resetowana w przeciwnym przypadku

N – resetowana

C- zawartość bitu 7 rejestru źródłowego

4.3.3.48. SRA (HL)

Opis: przesunięcie arytmetyczne wartości w pamięci adresowanej przez parę rejestrów HL w prawo o 1 bit. Zawartość bitu 0 jest kopiowana do flagi przeniesienia. Poprzednia zawartość bitu 7 pozostaje niezmieniona.

Kod instrukcji: 1 1 0 0 1 0 1 1 0 0 1 0 1 1 1 0

Identyfikacja rejestrów:

A – 111

B – 000 C – 001

D – 010 E – 011

H – 100 L – 101

Zmiana flag:

S – ustawiana jeżeli wynik jest ujemny, resetowana w przeciwnym przypadku

Z – ustawiana jeżeli wynikiem jest 0, resetowana w przeciwnym przypadku

H – resetowana

P/V – ustawiana jeżeli występuje parzysta liczba jedynek, resetowana w przeciwnym przypadku

N – resetowana

C- zawartość bitu 0 rejestru źródłowego

4.3.3.49. SRL (HL)

Opis: wartość w pamięci adresowana przez parę rejestrów HL jest przesuwana w prawo o 1 bit. Zawartość bitu 0 jest kopiowana do flagi przeniesienia, a bit 7 jest resetowany.

Kod instrukcji: 1 1 0 0 1 0 1 1 0 0 1 1 1 1 1 0

Identyfikacja rejestrów:

A – 111

B – 000 C – 001

D – 010 E – 011

H – 100 L – 101

Zmiana flag:

S – resetowana

Z – ustawiana jeżeli wynikiem jest 0, resetowana w przeciwnym przypadku

H – resetowana

P/V – ustawiana jeżeli występuje parzysta liczba jedynek, resetowana w przeciwnym przypadku

N – jest resetowana

C – dane z bitu 0 rejestru źródłowego

5. Analiza działania procesora

W niniejszej sekcji pokazane zostanie działanie naszego procesora. W kolejnych podsekcjach znajdują się przebiegi czasowe przykładowych instrukcji opisanych w poprzedniej części dokumentu.

5.1. Ładowanie danych do rejestrów

Poniższy program ładuje dane do rejestrów z wykorzystaniem różnych trybów adresowania.

Mnemonik	**Kod instrukcji w systemie szesnastkowym**
LD A, 1	3E 01
LD B, 2	06 02
LD (HL), 3	36 03
LD C, (HL)	4E
LD C, A	4f
LD (HL), A	77
LD B, (HL),	46

Powyższy kod oznacza wykonanie następujących operacji:

Załaduj wartość 1 do akumulatora.

Załaduj wartość 2 do rejestru B.

Załaduj wartość 3 do miejsca w pamięci adresowanego przez parę rejestrów HL.

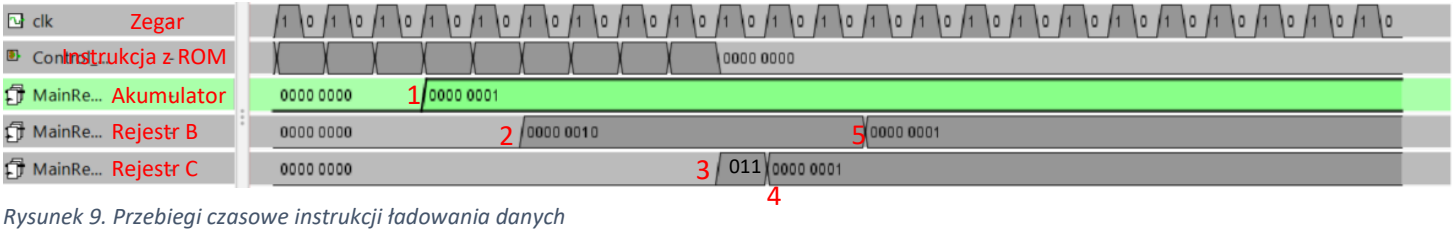
Załaduj wartość w pamięci adresowaną przez HL do rejestru C.

Załaduj wartość akumulatora do rejestru C.

Załaduj wartość akumulatora do miejsca w pamięci adresowanego przez parę rejestrów HL.

Załaduj wartość w pamięci adresowaną przez parę rejestrów HL do rejestru B.

Przebiegi czasowe:



Rysunek 9. Przebiegi czasowe instrukcji ładowania danych

Analiza przebiegu:

W celu dokładnego zobrazowania działania, po słownym opisie każdej operacji w nawiasie podana jest liczba, która odpowiada momentowi na wykresie czasowym oznaczonym tą samą liczbą.

Powyższy rysunek przedstawia zawartości akumulatora oraz rejestrów B i C w kolejnych cyklach zegara. Jak widać powyżej początkowo wartości 1 oraz 2 zostały wpisane kolejno do akumulatora i rejestru B(1 i 2). Następnie do rejestru C została wpisana wartość 3(3), która wcześniej została umieszczona w pamięci(stan pamięci nie jest pokazywany z racji na ograniczenia programu Logisim Evolution). Po tej operacji zawartość akumulatora została skopiowana do rejestru C(4).

Następnie zawartość akumulatora została wpisana do miejsca w pamięci adresowanego przez parę rejestrów HL, skąd została wpisana do rejestru B(5).

5.2. Operacje arytmetyczne i logiczne

Poniższy program prezentuje działanie kilku przykładowych operacji arytmetycznych.

Mnemonik	**Kod instrukcji w systemie szesnastkowym**
LD A, 4	3E 04
LD B, 8	06 08
ADD A, B	80
LD C, A	4F
OR 4	F6 0F
LD (HL), 1	36 01
SUB (HL)	96

Powyższy kod oznacza wykonanie następujących operacji:

Załaduj wartość 4 do akumulatora.

Załaduj wartość 8 do rejestru B.

Dodaj zawartości akumulatora i rejestru B, zapisz wynik w akumulatorze.

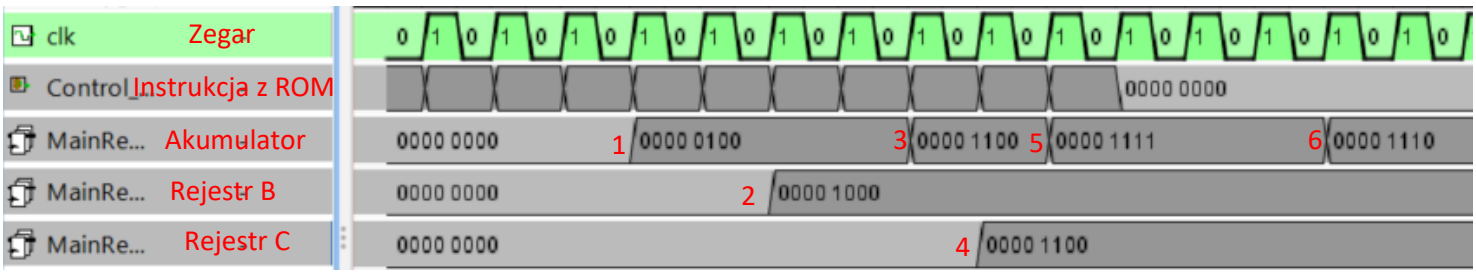
Skopiuj zawartość akumulatora do rejestru C.

Wykonaj logiczny OR dla zawartości akumulatora i argumentu natychmiastowego 4. Zapisz wynik w akumulatorze.

Załaduj wartość 1 do miejsca w pamięci adresowanego przez parę rejestrów HL.

Odejmij wartość w pamięci wskazywaną przez parę rejestrów HL od akumulatora. Zapisz wynik w akumulatorze.

Przebiegi czasowe:



Rysunek 10. Przebiegi czasowe operacji arytmetycznych

Analiza przebiegu:

W celu dokładnego zobrazowania działania, po słownym opisie każdej operacji w nawiasie podana jest liczba, która odpowiada momentowi na wykresie czasowym oznaczonym tą samą liczbą.

Powyższy rysunek przedstawia zawartość akumulatora oraz rejestrów B i C w kolejnych cyklach zegara. Na początku do akumulatora i rejestru B kolejno zostały załadowane wartości 4 i 8 przekazane jako argumenty natychmiastowe(1 i 2). Następnie zostało wykonane dodanie do siebie zawartości akumulatora i rejestru B, wynik został zapisany w akumulatorze(3). Potem zawartość akumulatora została skopiowana do rejestru C(4). W kolejnym kroku został wykonany logiczny OR pomiędzy zawartością akumulatora, a argumentem natychmiastowym 4, wynik operacji został zapisany w akumulatorze(5). W ostatnim kroku do miejsca w pamięci wskazywanego przez parę rejestrów HL został załadowany argument natychmiastowy 1(z racji na ograniczenia programu Logisim Evolution nie widać tego na przebiegach czasowych). W ostatnim kroku od akumulatora została odjęta wartość w pamięci wskazywana przez parę rejestrów HL, wynik został zapisany w akumulatorze(6).

5.3. Operacje przesunięć bitowych

Poniższy program prezentuje działanie kilku przykładowych instrukcji przesunięć bitowych.

Mnemonik	**Kod instrukcji w systemie szesnastkowym**
LD A, 255	3E FF
SLA A	CB 27
SLA A	CB 27
SRA A	CB 2F
SRL A	CB 3F

Powyższy kod oznacza wykonanie następujących operacji:

Załaduj wartość 255 do akumulatora.

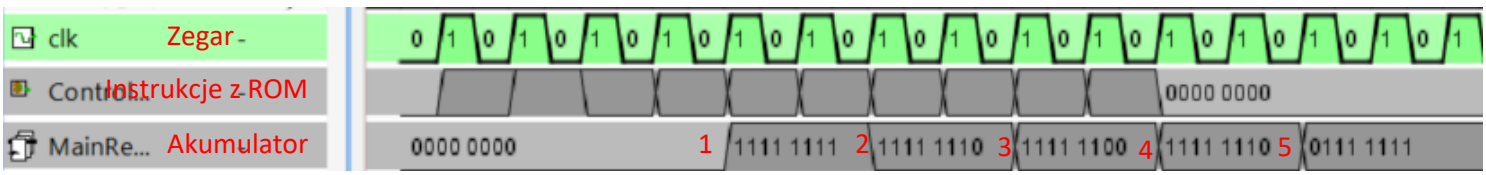
Wykonaj przesunięcie arytmetyczne w lewo o 1 bit.

Wykonaj przesunięcie arytmetyczne w lewo o 1 bit.

Wykonaj przesunięcie arytmetyczne w prawo o 1 bit.

Wykonaj przesunięcie logiczne w prawo o 1 bit.

Przebiegi czasowe:



Rysunek 11. Przebiegi czasowe operacji przesunięć bitowych

Analiza przebiegu:

W celu dokładnego zobrazowania działania, po słownym opisie każdej operacji w nawiasie podana jest liczba, która odpowiada momentowi na wykresie czasowym oznaczonym tą samą liczbą.

Powyższy rysunek przedstawia zawartość akumulatora w kolejnych cyklach zegara. Na początku do akumulatora ładowana jest wartość 255(1). Następnie wykonywana jest operacja przesunięcia arytmetycznego w lewo o 1 bit(2). W kolejnym kroku operacja ta jest powtarzana(3). Następną instrukcją jest przesunięcie arytmetyczne w prawo o 1 bit(4). Ostatnim krokiem jest przesunięcie logiczne w prawo o 1 bit(5).

5.4. Skok bezwarunkowy

Poniższy program prezentuje działanie skoku bezwarunkowego

Mnemonik	**Kod instrukcji w systemie szesnastkowym**
LD A, 3	3E 03
DEC A	3D
JP 00 02(adres w pamięci ROM)	C3 02 00

Powyższy krok oznacza wykonanie następujących operacji:

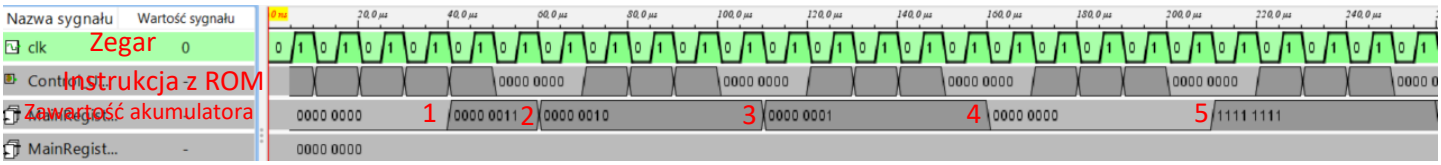
Załaduj wartość 3 do akumulatora.

Dekrementuj akumulator.

Skocz do miejsca w pamięci ROM, gdzie znajduje się instrukcja DEC A

W tym momencie warto zaznaczyć, że przy podawaniu adresu skoku należy najpierw podać młodszy bajt adresu.

Przebiegi czasowe:



Rysunek 12. Przebiegi czasowe instrukcji skoku

Analiza przebiegu:

W celu dokładnego zobrazowania działania, po słownym opisie każdej operacji w nawiasie podana jest liczba, która odpowiada momentowi na wykresie czasowym oznaczonym tą samą liczbą.

Powyższy rysunek przedstawia zawartość akumulatora w kolejnych cyklach zegara. Na początku do akumulatora ładowana jest wartość 1(1). Następnie dekrementujemy akumulator(2).Po tym następuje instrukcja skoku do miejsca w pamięci, w którym zapisana jest instrukcja

dekrementacji akumulatora. Akumulator jest zatem ponownie dekrementowany(3). W ten sposób akumulator będzie nieprzerwanie dekrementowany(4 i 5).

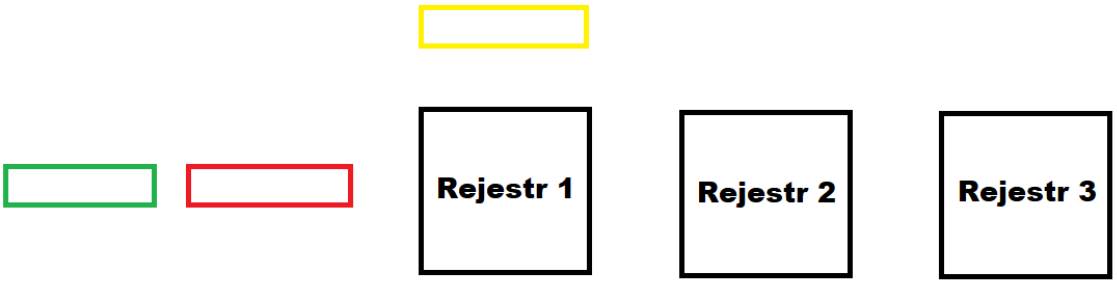
5.5. Proces dekodowania instrukcji

Po pobraniu instrukcji z pamięci ROM jest ona transportowana do jednostki sterującej naszego procesora (ang. Control Unit, CU). Tam instrukcja jest dekodowana i wysyłane są odpowiednie sygnały sterujące dalszymi procesami w układzie.

Nasza realizacja jednostki Control Unit składa się z trzech 8-bitowych rejestrów, co pozwala na odczytywanie instrukcji jedno, dwu i trzy bajtowych. Proces ładownia instrukcji do tych trzech rejestrów prezentuje się następująco:

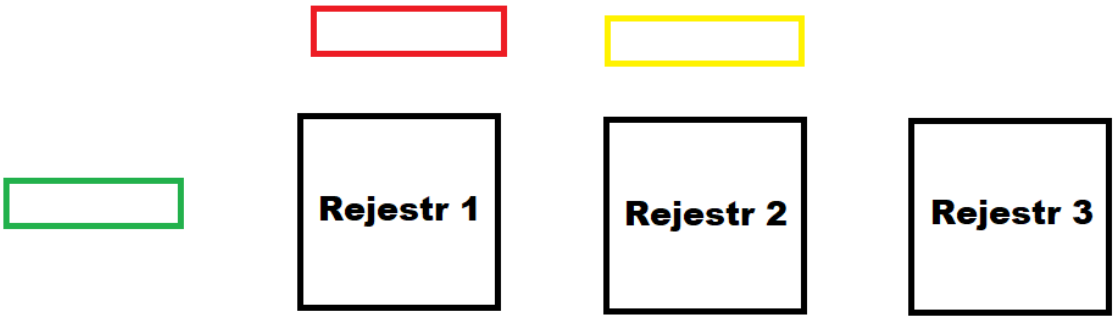
Rysunek 13. Graficzna prezentacja 3 bajtów

Powyższy rysunek w graficzny sposób prezentuje instrukcję 3 bajtową, gdzie każdy bajt oznaczony jest przez prostokąt w innym kolorze. Prostokąt żółty oznacza bajt najmłodszy, natomiast prostokąt zielony, bajt najstarszy.



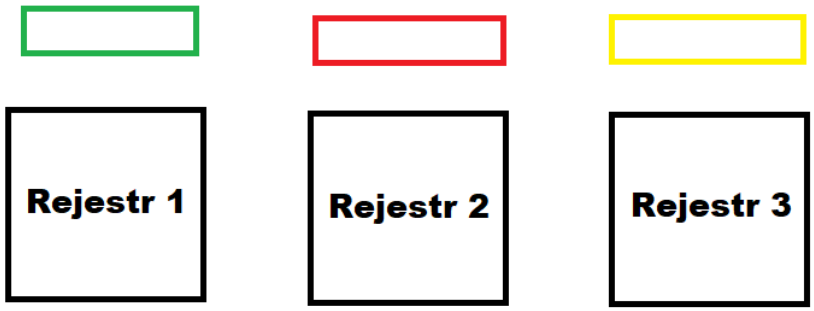
Rysunek 14. Dekodowanie instrukcji, krok 1.

W pierwszym cyklu zegara najmłodszy bajt instrukcji ładowany jest do rejestru 1



Rysunek 15. Dekodowanie instrukcji, krok 2.

W drugim cyklu zegara bajt załadowany do rejestru 1 jest transferowany do rejestru 2, a kolejny bajt instrukcji jest ładowany do rejestru 1.



Rysunek 16. Dekodowanie instrukcji, krok 3.

W trzecim cyklu zegara bajt z rejestru 2 jest transportowany do rejestru 3, bajt z rejestru 1 do rejestru 2, a ostatni bajt instrukcji ładowany do rejestru 1.

Powyższy mechanizm oznacza, że czytając instrukcję 1 bajtową, jednostka sterująca potrzebuje 3 cykli zegara, by umieścić instrukcje w odpowiednim rejestrze. Dopiero po tym procesie następuje właściwe dekodowanie instrukcji i wysyłanie odpowiednich sygnałów w układzie.

Sam mechanizm dekodowania został zaprojektowany w oparciu o zależności pomiędzy kodami instrukcji. Jako przykład posłużymy się tutaj instrukcjami arytmetycznymi.

Kody instrukcji podstawowych operacji arytmetycznych i logicznych prezentują się następująco:

ADD A, r – 1 0 0 0 0 r r r

ADC A, r – 1 0 0 0 1 r r r

SUB r – 1 0 0 1 0 r r r

SBC r – 1 0 0 1 1 r r r

AND r – 1 0 1 0 0 r r r

XOR r – 1 0 1 0 1 r r r

OR r – 1 0 1 1 0 r r r

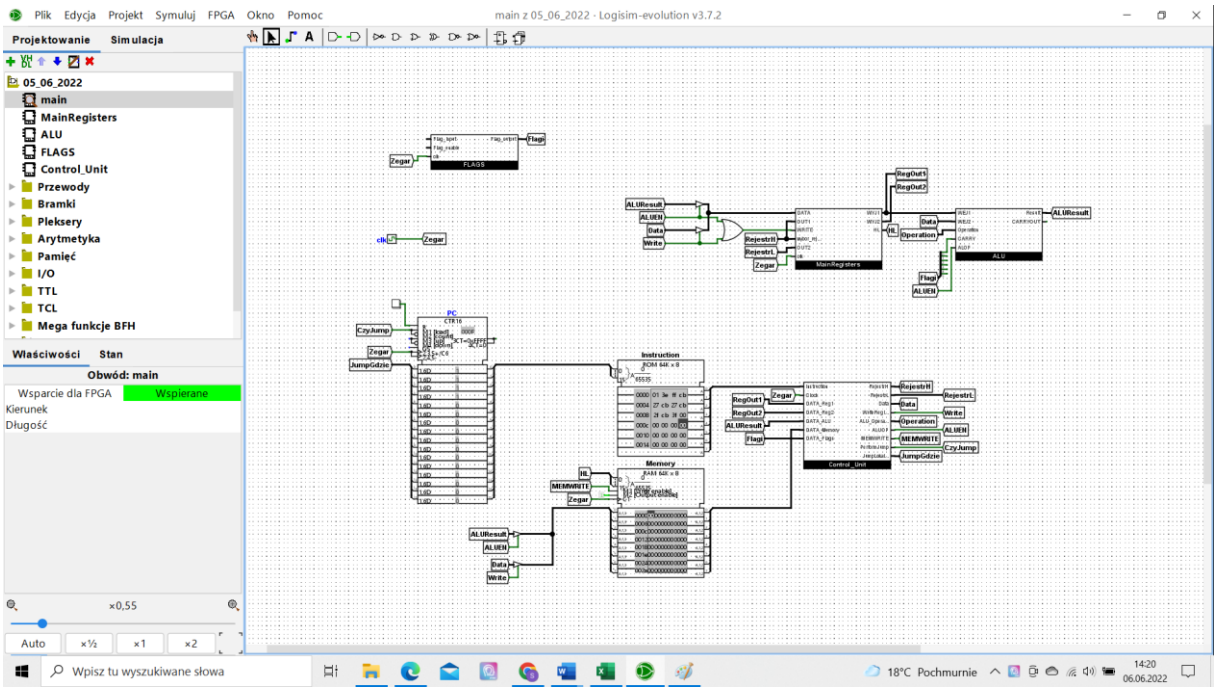
Jak widać powyżej, kody te można ułożyć w kolejności rosnącej, w której każdy kolejny kod jest o 1 większy od poprzedniego.

Nasza jednostka sterująca po rozpoznaniu dwóch najstarszych bitów jako 1 oraz 0 wie, że będzie wykonywać operację arytmetyczną lub logiczną. Następnie kolejne 3 bity określają rodzaj wykonywanej operacji. Jak można zauważyć na Rysunku 4 przedstawiającym nasze ALU, bloki odpowiedzialne za konkretne operacje zostały ułożone w dokładnie tej samej kolejności, jak posortowane powyżej kody instrukcji. Dzięki takiemu układowi bity 5, 4 i 3 są przesyłane bezpośrednio do multipleksera, który w ALU jest odpowiedzialny za wybór operacji.

6. Instrukcja obsługi procesora w programie Logisim Evolution

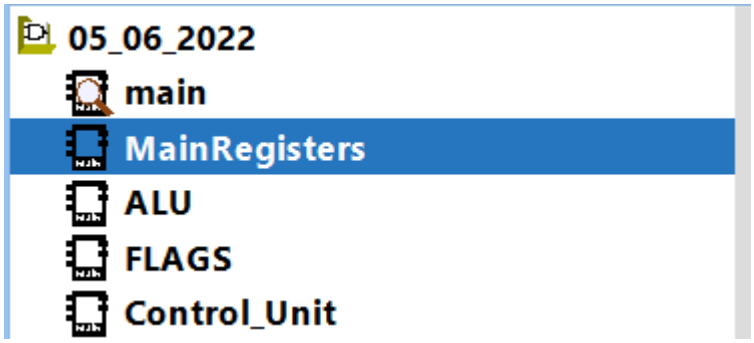
Niniejsza sekcja opisuje w jaki sposób użytkownik może testować działanie naszej implementacji procesora Z80.

Po uruchomieniu programu Logisim Evolution powinien ukazać się schemat całego układu procesora.



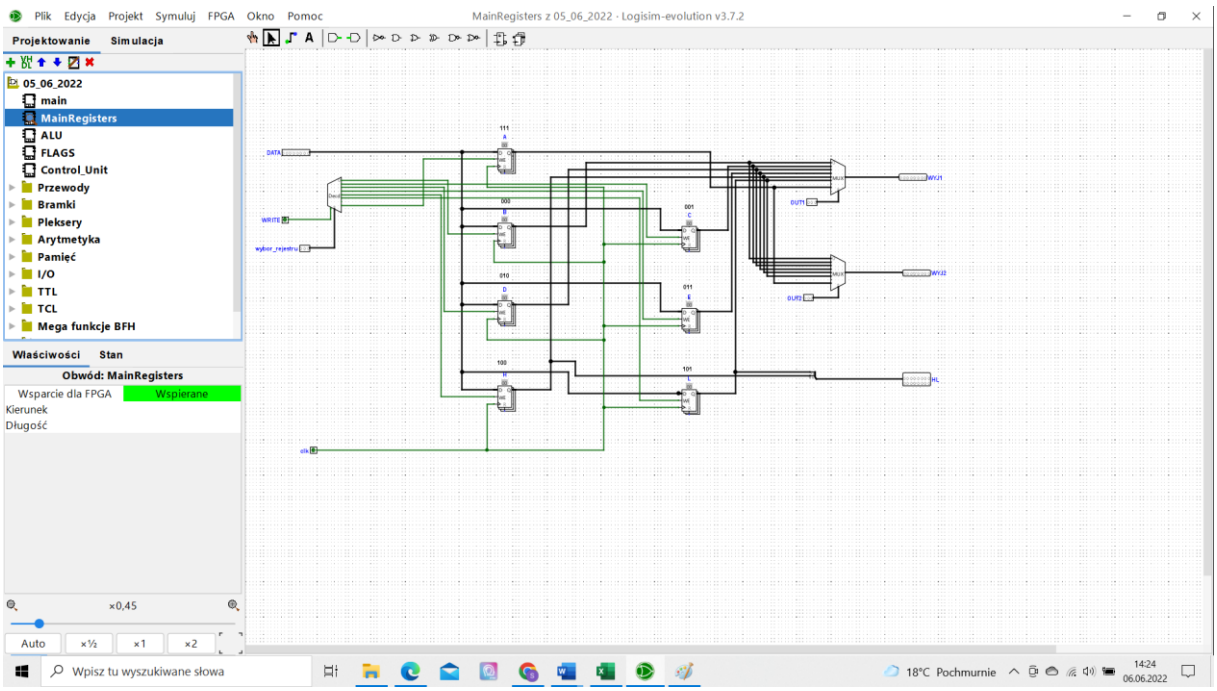
Widok 1 .Ekran główny programu

W lewym górnym rogu programu widoczny jest spis wszystkich stworzonych części naszego układu. Po kliknięciu dwukrotnie, w wybraną część, program przeniesie użytkownika do widoku obrazującego budowę wybranego elementu.



Widok 2. Elementy układu

Dla przykładu podwójne kliknięcie w „MianRegisters” sprawi, że widok główny naszego programu będzie się prezentował następująco:

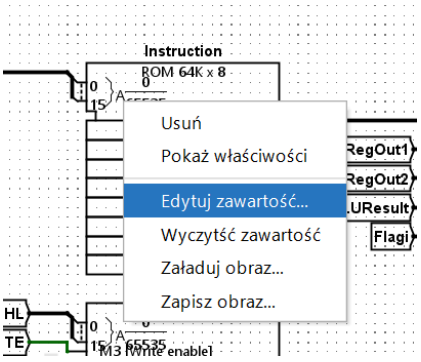


Widok 3. Schemat wybranego elementu

Podwójne kliknięcie w „main” sprawi, że wrócimy z powrotem do widoku całego układu.

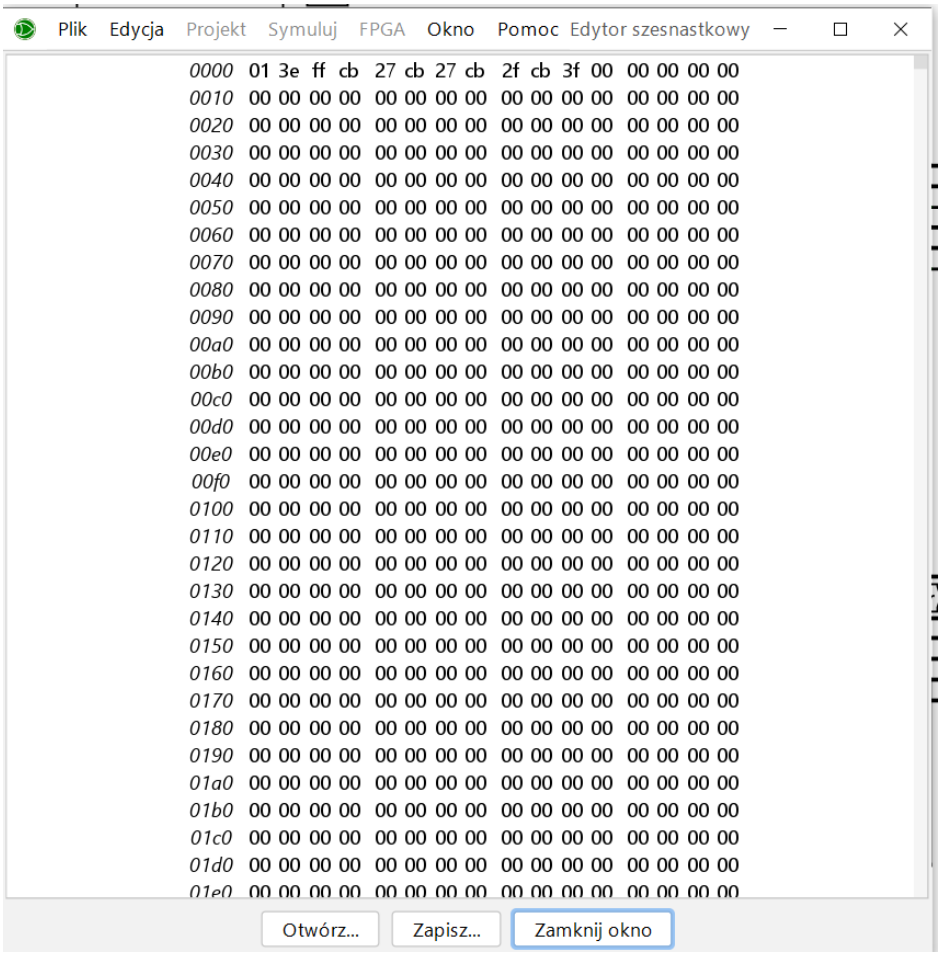
Chcąc sprawdzić działanie naszego procesora, musimy na początku wgrać kody instrukcji w szesnastkowym systemie liczbowym do pamięci ROM.

Aby to zrobić należy kliknąć prawym przyciskiem myszy na element nazwany „Instruction”. Po tej akcji na ekranie wyświetli się lista opcji do wybrania. Należy kliknąć w opcję „Edytuj zawartość”



Widok 4. Edytowanie zawartości pamięci ROM

Na ekranie wyświetli się tabela reprezentująca zawartość pamięci ROM.



Widok 5. Zawartość pamięci ROM

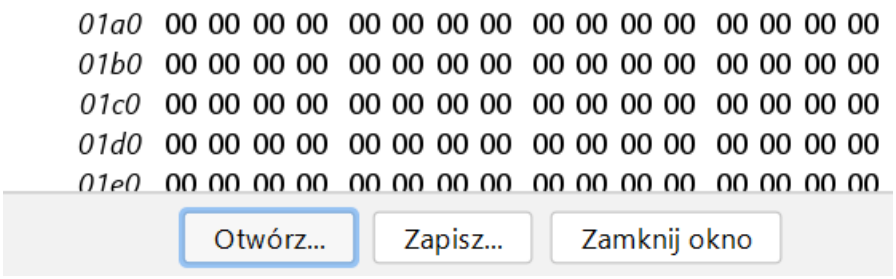
W celu wpisania kodu instrukcji należy kliknąć myszką w odpowiednią parę zer, i wpisać kod instrukcji z klawiatury.

Po wybraniu danej pary zer, po kolejnych parach można się poruszać za pomocą strzałek na klawiaturze.

Drugim sposobem wgrania instrukcji do pamięci ROM jest wczytanie jej z pliku tekstowego. Aby to zrobić należy mieć przygotowany plik tekstowy, w którym będą zapisane kody instrukcji w szesnastkowym systemie liczbowym, każdy bajt musi być oddzielony spacją.

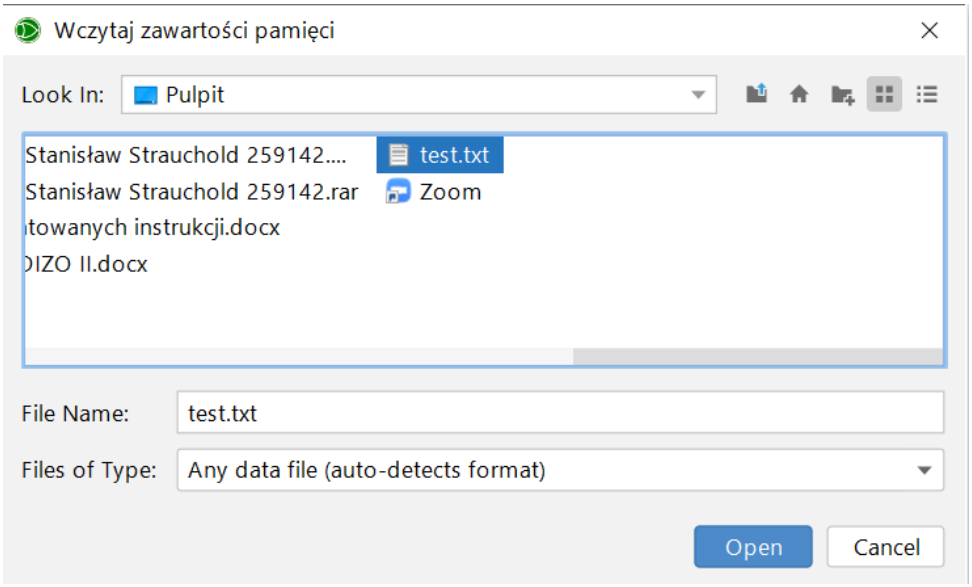
Widok 6. Zawartość pliku tekstowego z instrukcjami

Następnie po wyświetleniu się tabeli reprezentującej zawartość pamięci ROM należy kliknąć przycisk „Otwórz…”



Widok 7. Import pliku tekstowego 1

Po kliknięciu ukaże nam się okno, w którym należy przejść do lokalizacji, w której zapisaliśmy nasz plik tekstowy.

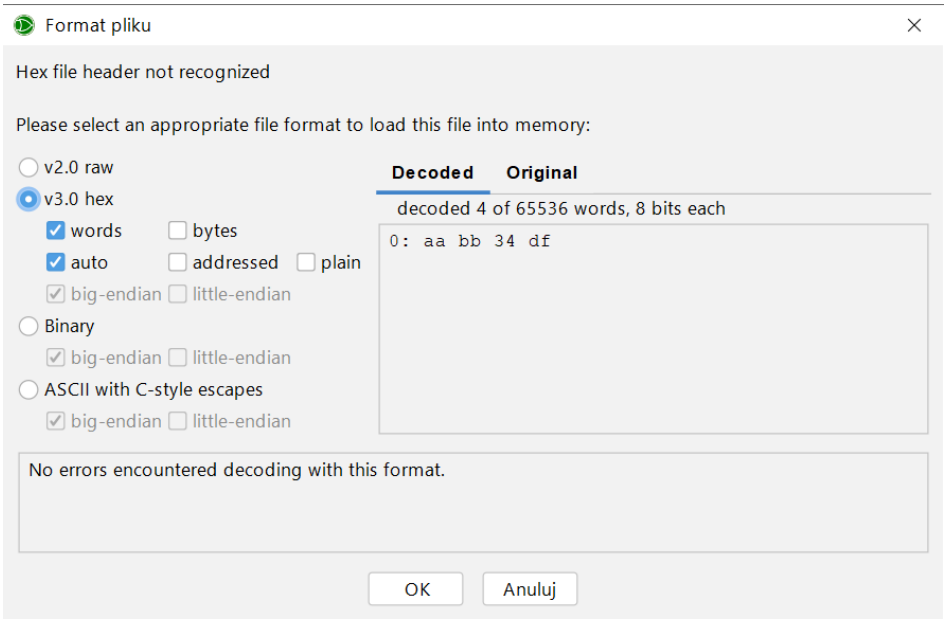


Widok 8. Import pliku tekstowego 2

Nasz plik tekstowy był zapisany na pulpicie, więc w polu „Look In:” wybraliśmy Pulpit. Po przejściu do odpowiedniego folderu, aby znaleźć plik tekstowy należy użyć poziomego suwaka widocznego w oknie oznaczonym niebieską ramką. Po znalezieniu naszego pliku należy go zaznaczyć, klikając na niego lewym przyciskiem myszy, a następnie kliknąć przycisk „Open” zaznaczony na powyższym rysunku na niebiesko.

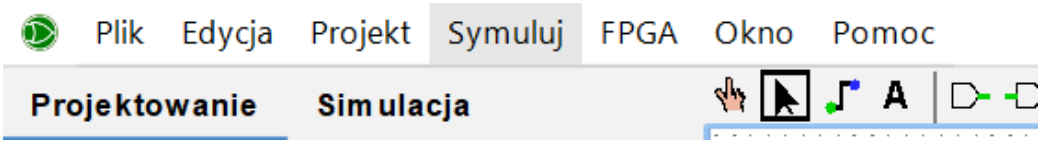
Po kliknięciu przycisku „Open” ukaże się widoczne poniżej okno. Po jego lewej stronie należy wybrać ustawienia takie same jak Widoku 9. Zagwarantuje to rozpoznanie elementów pliku tekstowego jako liczby w systemie szesnastkowym.

Po zaznaczeniu odpowiednich opcji należy kliknąć przycisk „Ok”. Dane będą załadowane do pamięci ROM.



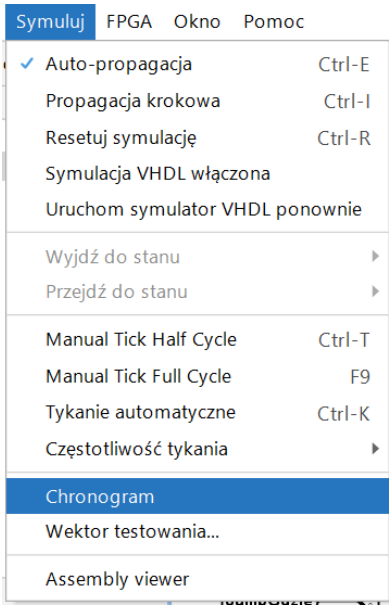
Widok 9.. Import pliku tekstowego 3

Po wczytaniu danych warto włączyć generowanie przebiegów czasowych aby móc śledzić zmiany sygnałów w układzie w trakcie działania programu. W tym celu, w głównym oknie programu należy kliknąć przycisk „Symuluj”, który znajduje się w górnym pasku aplikacji.



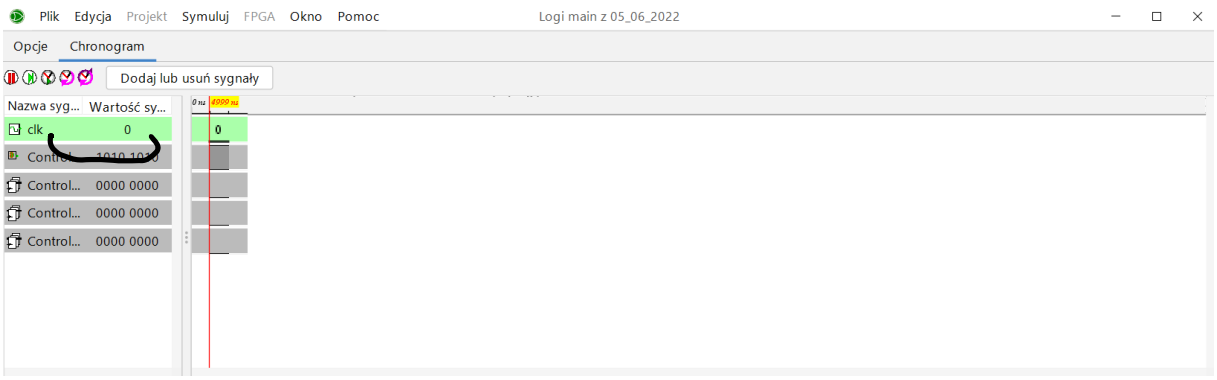
Widok 10. Włączenie symulacji 1

Po kliknięciu rozwinie się menu, w którym należy wybrać opcję „Chronogram”



Widok 11. Włączenie symulacji 2

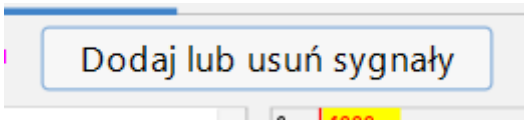
W wyświetlonym oknie należy ponownie wybrać opcję Chronogram



Widok 12. Włączenie symulacji 3

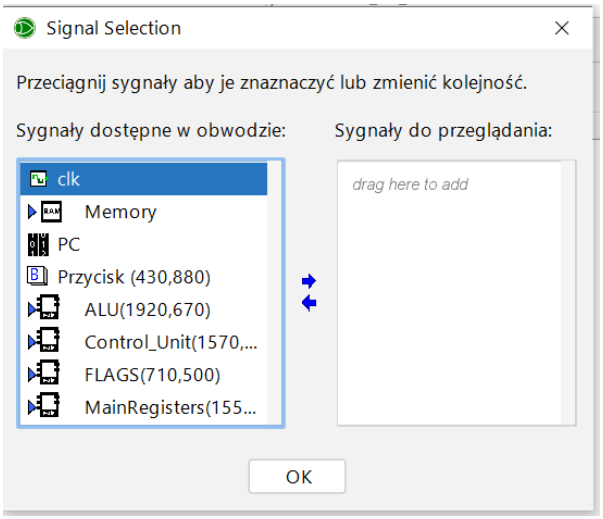
Powinno ukazać się okno podobne do powyższego rysunku. Jedyną różnicą może być ilość sygnałów zaznaczonych na rysunku na szaro i zielono.

W celu dodania sygnałów (lub ich usunięcia) do przebiegu czasowego należy kliknąć w przycisk „Dodaj lub usuń sygnały".



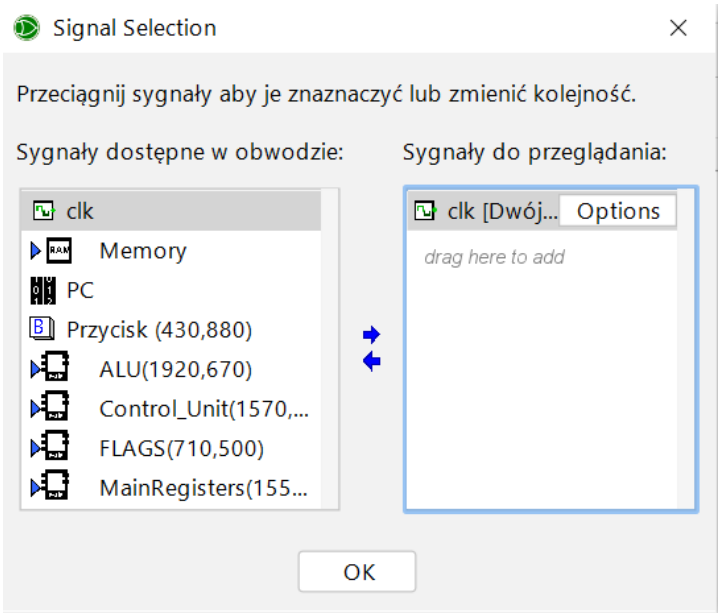
Widok 13. Włączenie symulacji 4

Aby dodać sygnał należy najpierw wybrać go z okna po lewej, klikając na niego lewym przyciskiem myszy, a następnie kliknąć strzałkę wskazującą w prawo w środkowej części. Sygnał powinien ukazać się w prawym oknie.



Widok 14. Włączenie symulacji 5

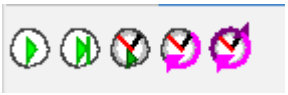
Analogicznie aby usunąć sygnał należy kliknąć na niego w prawym oknie, a następnie kliknąć strzałkę skierowaną w lewą stronę.



Widok 15. Włączenie symulacji 6

Po wybraniu odpowiednich sygnałów należy kliknąć przycisk „Ok”. Powinniśmy wrócić do okna przedstawiającego wybrane sygnały.

Kolejnym krokiem jest uruchomienie symulacji. W tym celu należy użyć następujących przycisków:



Widok 16. Włączenie symulacji 7

Klikając przycisk środkowy uruchomimy automatyczne tykanie zegara w trakcie trwania symulacji. Aby włączyć symulację należy kliknąć przycisk pierwszy od lewej. W tym momencie powinniśmy widzieć zmiany sygnałów oraz cykle zegara. Symulację można w każdej chwili zatrzymać i włączyć z powrotem używając przycisku po lewej stronie.

Przed uruchomieniem nowej symulacji należy wyczyścić przebiegi czasowe klikając przycisk „Resetuj sumulację”. Do którego można się dostać w sposób widoczny na zdjęciu poniżej.

Widok 17. Resetowanie symulacji

7. Podsumowania i wnioski

W tej sekcji opisane zostanie podsumowanie naszej pracy oraz wnioski, do których doszliśmy po analizie wykonanego projektu.

7.1. Organizacja pracy

Z racji, że zajęcia projektowe z przedmiotu Architektura Komputerów 2 odbywały się co 2 tygodnie, postępy naszej pracy również były zauważalne co taki okres czasu. Początkowo nasza uwaga skupiona była na zapoznaniu się z architekturą procesora Z80 oraz na poznaniu programu Logisim(nie Logisim Evolution). Pierwszym krokiem implementacji procesora była próba odtworzenia grupy rejestrów ogólnego przeznaczenia oraz zaprojektowanie mechanizmu wczytywania do tychże rejestrów danych. Następnie dodawane zostały mechanizmy kopiowania danych z rejestru do rejestru, a następnie przekazywania danych z wybranych do rejestrów do innych elementów układu. Po wykonaniu tych kroków zmieniliśmy program Logisim na Logisim Evolution, gdyż w tym pierwszym nie ma możliwości generowania wykresów czasowych. Po tej zmianie naszym kolejnym celem było zbudowanie jednostki arytmetyczno logicznej, która przyjmowała by dane z rejestrów, a następnie po wybraniu odpowiedniej operacji, przechodziła do działania. W kolejnym kroku zbudowaliśmy rejestr flag, a następnie przeszliśmy do prób zautomatyzowania działania procesora tak, by po podaniu odpowiedniego kodu operacji, ta sama się wykonywała. W tym celu dodaliśmy do układu takie elementu jak licznik rozkazów, pamięć ROM oraz pamięć RAM. Następnie przeszliśmy do projektowania jednostki sterującej, która była głównym tematem naszej pracy niemal do końca robienia projektu.

Podział obowiązków w grupie prezentował się następująco. Początkowe etapy były robione wspólnie, wszystkie problemy były na bieżąco dyskutowane. Następnie w miarę poznania mocnych stron członków zespołu, prace zostały podzielone. Co etap dyskutowane były problemy, które wymagały rozwiązania, następnie następował podział obowiązków i obie strony wykonywały przydzielone zadania. W przypadku napotkania przez któregoś z członków zespołu trudności, problem był rozwiązywany wspólnie. Dopiero w ostatnim etapie nastąpił wyraźniejszy podział obowiązków, mianowicie Michał Tłuczek zajął się optymalizacją i rozbudowaniem jednostki sterującej, natomiast Stanisław Strauchold skupił się na dokumentacji. Pomimo mocno odmiennych zadań strony dyskutowały każdy problem wspólnie i razem go rozwiązywały.

7.2. Napotkane problemy

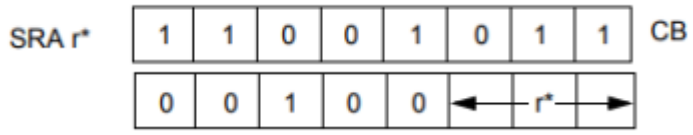
Podczas wykonywania projektu spotkaliśmy się z kilkoma, różnego rodzaju problemami. Każdy z nich zostanie osobno opisany.

7.2.1. Brak wykresów czasowych w programie Logisim

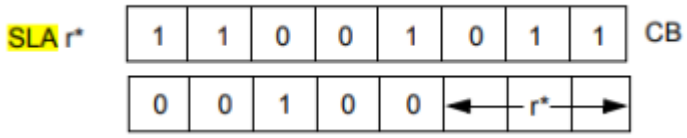
Pierwszym napotkanym przez nas problemem był fakt, że program Logisim, nie posiada funkcji rysowania wykresów czasowych. Z tego względu zmuszeni byliśmy odtworzyć nasz ówczesny etap prac w programie Logisim Evolution, który jest bardzo podobny do programu Logisim. Niestety próba otwarcia pliku stworzonego w Logisim'ie w Logisim'ie Evolution nie przynosiła zamierzonych rezultatów, gdyż elementy układu nie zgadzały się z pierwotnym ułożeniem. Z tego powodu musieliśmy ręcznie odbudować cały układ w nowym programie.

7.2.2. Błędy w dokumentacji

Kolejnym problemem, który został wykryty w ostatniej chwili były błędy w dokumentacji. W trakcie tworzenia projektu korzystaliśmy z dwóch dokumentacji, które wydały nam się najbardziej przejrzyste. Pod sam koniec na etapie testów okazało się, że w jednej z nich są bardzo znaczące błędy. Mianowicie nie zgadzały kody instrukcji. Jako przykład pokażemy kody dwóch różnych instrukcji w drugiej dokumentacji:



Błąd 1. Kod instrukcji SRA



Błąd 2. Kod instrukcji SLA

Jak widać powyżej dokumentacja podaje, że obie operacje mają ten sam kod instrukcji.

Ze względu na ten błąd zmuszeni byliśmy sprawdzać poprawność wszystkich zaimplementowanych operacji oraz zmieniać zawartość tabelek z kodami instrukcji pokazanych na początku dokumentacji.

7.3. Wnioski

Celem naszego projektu była własna implementacja procesora Z80. Układ miał mieć architekturę zbliżoną do architektury Z80 oraz wykonywać takie same instrukcje, które miały być kodowane tymi samymi kodami. Stworzony przez nas układ nie jest dokładnym odwzorowaniem procesora Z80, natomiast jego główne elementy są zgodne z oryginałem. Ponadto udało się zaimplementować kilkadziesiąt instrukcji, których działanie jest takie same jak w oryginalnym procesorze. W trakcie prac nad projektem dość szczegółowo zapoznaliśmy się ze sposobem działania procesorów i wykonywania przez ich instrukcji. Projekt ten był doskonałym uzupełnieniem wykładu z przedmiotu Architektura Komputerów 2, gdyż wiedzę zdobytą na tym wykładzie mogliśmy wykorzystać w praktyce. Wykonany przez nas procesor jest w stanie wykonywać proste programy, które zawierają w sobie instrukcje manipulacji danymi, arytmetyczne oraz logiczne. Zaawansowanie tychże programów jest zwiększone poprzez możliwość wykonywania instrukcji skoków i porównań.

8. Bibliografia

[1.] „Z80 Family CPU User Manual” Zilog Worldwide Headquarters, 532 Race Street, San Jose, dokument PDF

[2.] Rodnay Zaks , „Programming the Z80, 3rd Edition” , SYBEX, 1981

[3.] Strona opisująca architekturę Z80 https://landley.net/history/mirror/cpm/z80.html

Sekcja dodana po sprawdzeniu poprzedniej wersji raportu przez prowadzącego. Znajduje się poniżej całego dokumentu by nie psuć jego struktury oraz spisu treści.

Aby pokazać działanie różnych instrukcji w naszym procesorze, przygotowaliśmy przykładowy program, którego działanie było następujące:

- wczytaj argument natychmiastowy 2 do akumulatora

- dekrementuj A

- Jeżeli A nie wynosi 0, powtórz poprzedni krok

- dodaj 8 do zawartości akumualtora

- przesunięcie arytmetyczne zawartości akumulatora w lewo o 1 bit

- skopiuj zawartość akumulatora do rejestru B

- odejmij zawartość rejestru B od zawartości akumulatora, zapisz wynik w akumulatorze

- jeżeli wynik nie równa się 0, wróć do operacji kopiowania zawartości akumulatora do rejestru B

- załaduj argument natychmiastowy 8 do miejsca w pamięci adresowanego przez parę rejestrów HL

- inkrementuj wartość w pamięci wskazywaną przez parę rejestrów HL

- załaduj wartość w pamięci wskazywaną przez parę rejestrów HL do akumulatora

- dodaj wartość w pamięci wskazywaną przez parę rejestrów HL do zawartości akumulatora, zapisz wynik w akumulatorze

Zapis bajtowy tego programu prezentuje się następująco:

3E 02 3D C2 01 00 C6 08 CB 27 47 90 C2 05 00 36 08 34 7E 86

Przebieg czasowy wykonywanej operacji