Jakub Sobieszek
232838
TN Czw 11:15

06.06.2019
Wrocław

Politechnika Wrocławska
Wydział Elektroniki

Organizacja i Architektura Komputerów

Projekt i implementacja mikrokontrolera intel 8051

Dokumentacja

Prowadzący:
Dr inż. Tadeusz Tomczak

1. Cele i założenia projektu

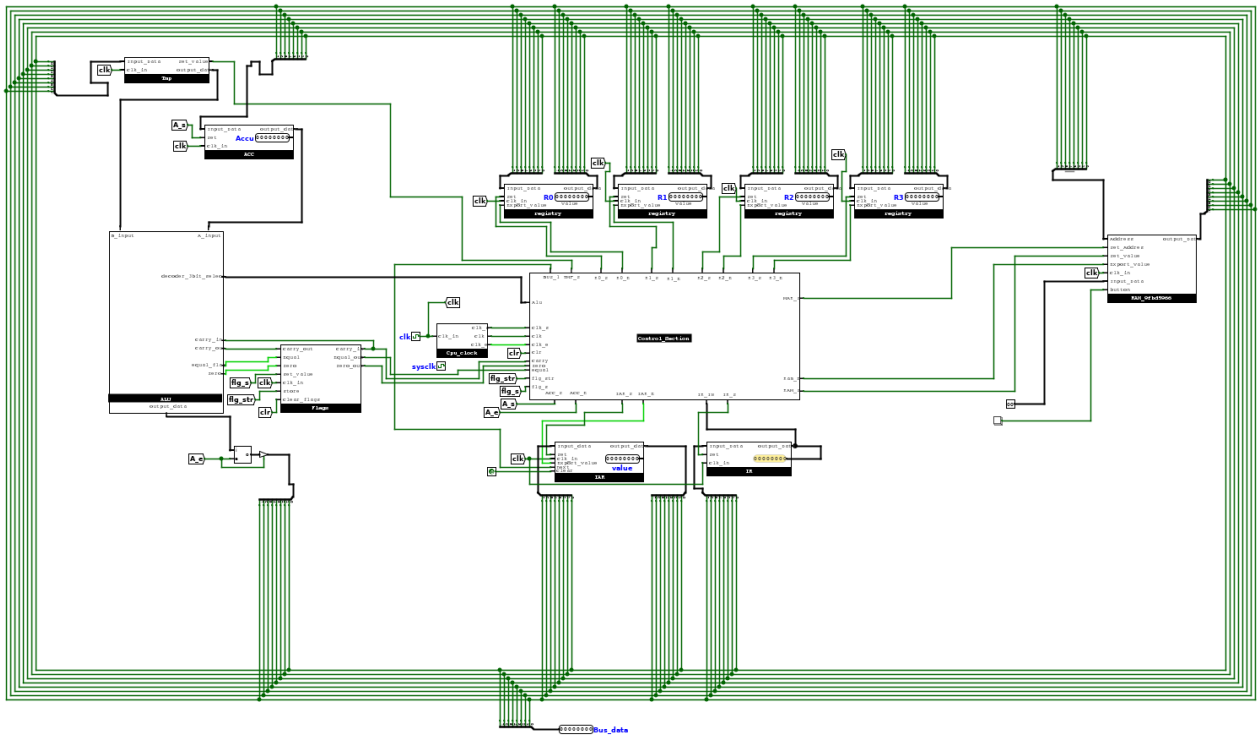
Celem projektu było wykonanie, za pomocą programu do projektowania układów cyfrowych, kopii mikrokontrolera intel 8051 z roku 1980. Należało stworzyć układ zawierający w sobie jednostkę arytmetyczno-logiczną, magistralę, jednostkę sterującą, bank rejestrów, rejestr flag, oraz dodatkowe mniejsze elementy potrzebne do działania układu. W założeniu konstrukcja miała używać oryginalnych kodów operacji w/w mikrokontrolera, umożliwiać wczytywanie, przenoszenie danych, wykonywanie operacji arytmetyczno-logicznych, oraz skoków.

2. Realizacja

Układ powstał z wykorzystaniem narzędzia Logisim. Początkowo w jego starszej wersji bez możliwości generowania wykresów czasowych. Używany do tego był napisany własnoręcznie skrypt w języku Python. W późniejszym etapie narzędzie to zastąpione zostało przez Logisim-evolution, projekt kontynuujący, nierozwijaną już oryginalną wersję programu.

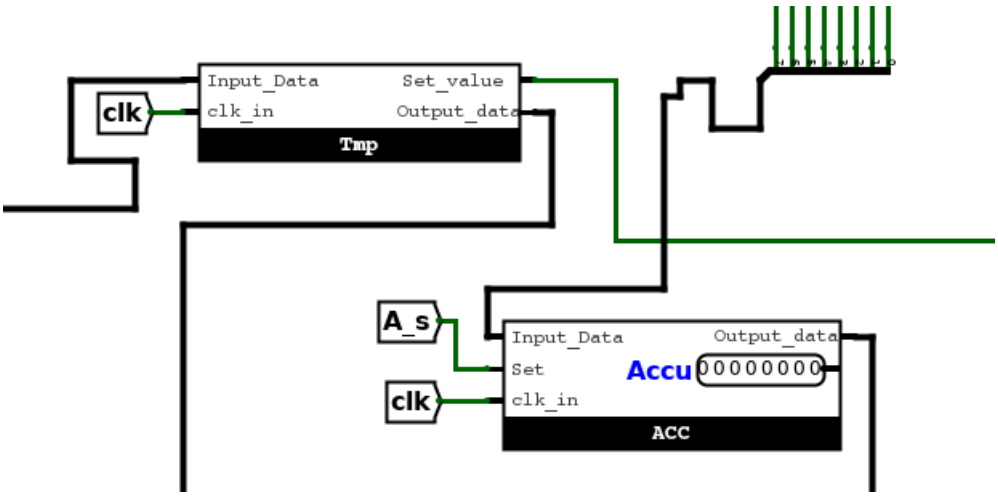
Udało się zrealizować opisane powyżej założenia. Stworzona konstrukcja funkcjonuje poprzez realizowanie kolejnych kodów operacji umieszczonych przez użytkownika w wewnętrznym RAM’ie kontrolera. W przypadku skoków następuje przejście na wskazany adres. Zawartość RAM’u można zmodyfikować wewnątrz programu Logisim, a następnie zapisać i eksportować tworząc swego rodzaju program. Następnie programy można ładować do komponentu aby móc je wykonać. Trzymając się odpowiedniej konwencji, można też program napisać od razu w pliku tekstowym. Program posiada formę kodów operacji w systemie szesnastkowym. Pisząc więc program, użytkownik odgrywa rolę programisty oraz kompilatora.

3. Architektura



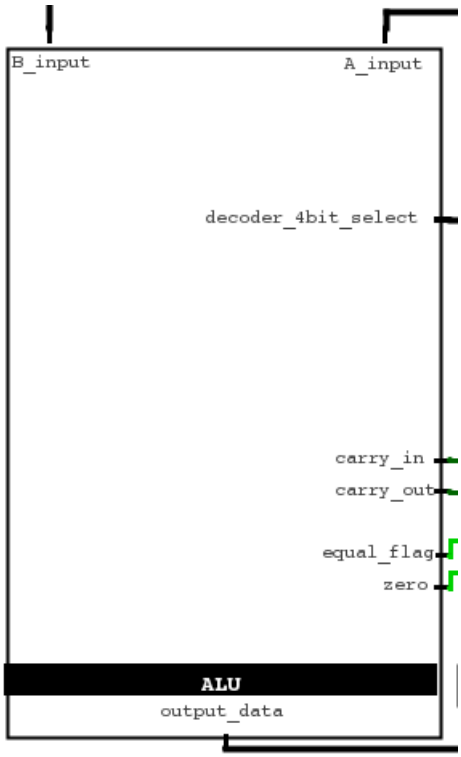
Spis komponentów:
-Akumulator
-Rejestr tymczasowy
-Jednostka Arytmetyczno-Logiczna
-Rejestr Flag
-Bank 4 rejestrów
-Magistrala
-IRAM
-Rejestr Adresów Instrukcji
-Rejestr Instrukcji
-Jednostka sterująca
-Zegar
-Generator sygnałów „Set” i „Export”

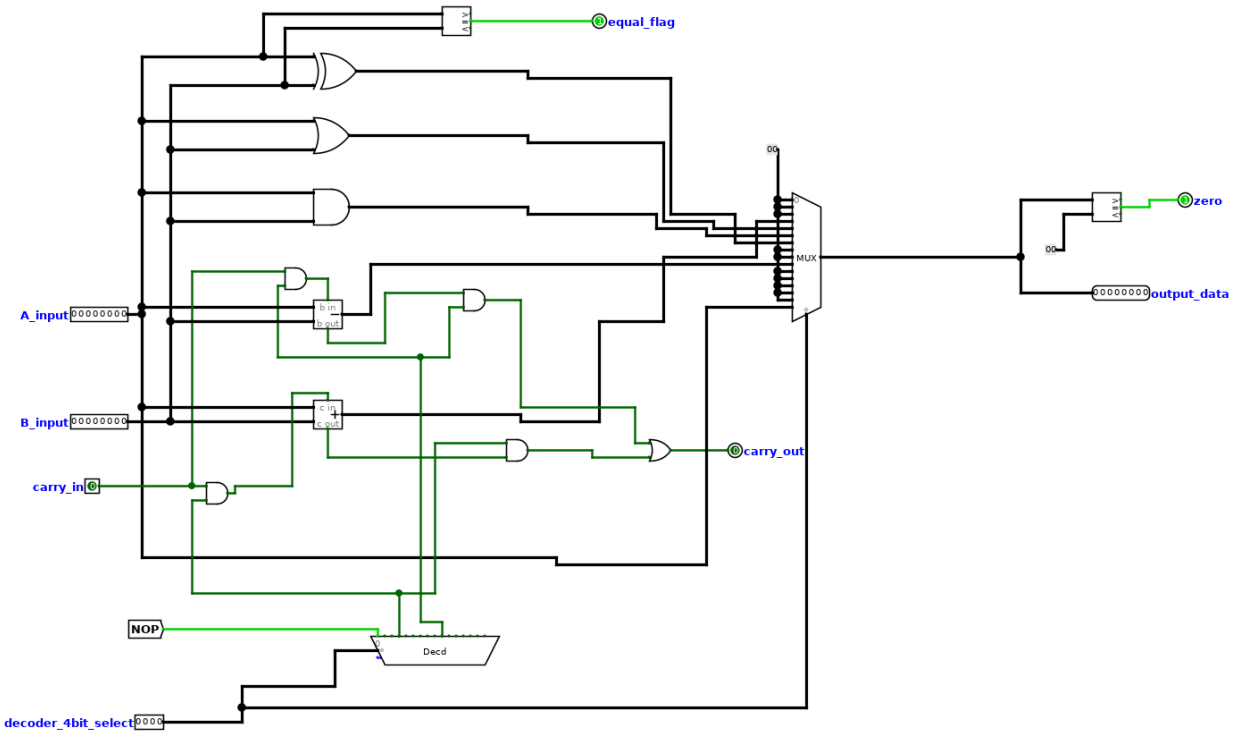
Akumulator i rejestr tymczasowy



Akumulator (*ACC*) to rejestr wykorzystywany w operacjach jednostki Arytmetyczno-Logicznej. Zapisywany jest też w nim wynik operacji. Rejestr tymczasowy (*Tmp*) służy jako drugi argument jednostki arytmetyczno logicznej.
Dla przykładu: wykonując operację ADD A, R0 (dodanie wartości w rejestrze 0 do wartości akumulatora), zawartość rejestru R0 zostanie skopiowana do rejestru tymczasowego, a następnie zostanie wykonanie dodanie jej i zawartości Akumulatora, a wynik operacji nadpisze wartość w Akumulatorze.

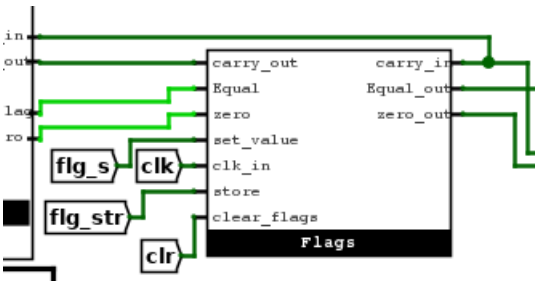
Jednostka Arytmetyczno-Logiczna

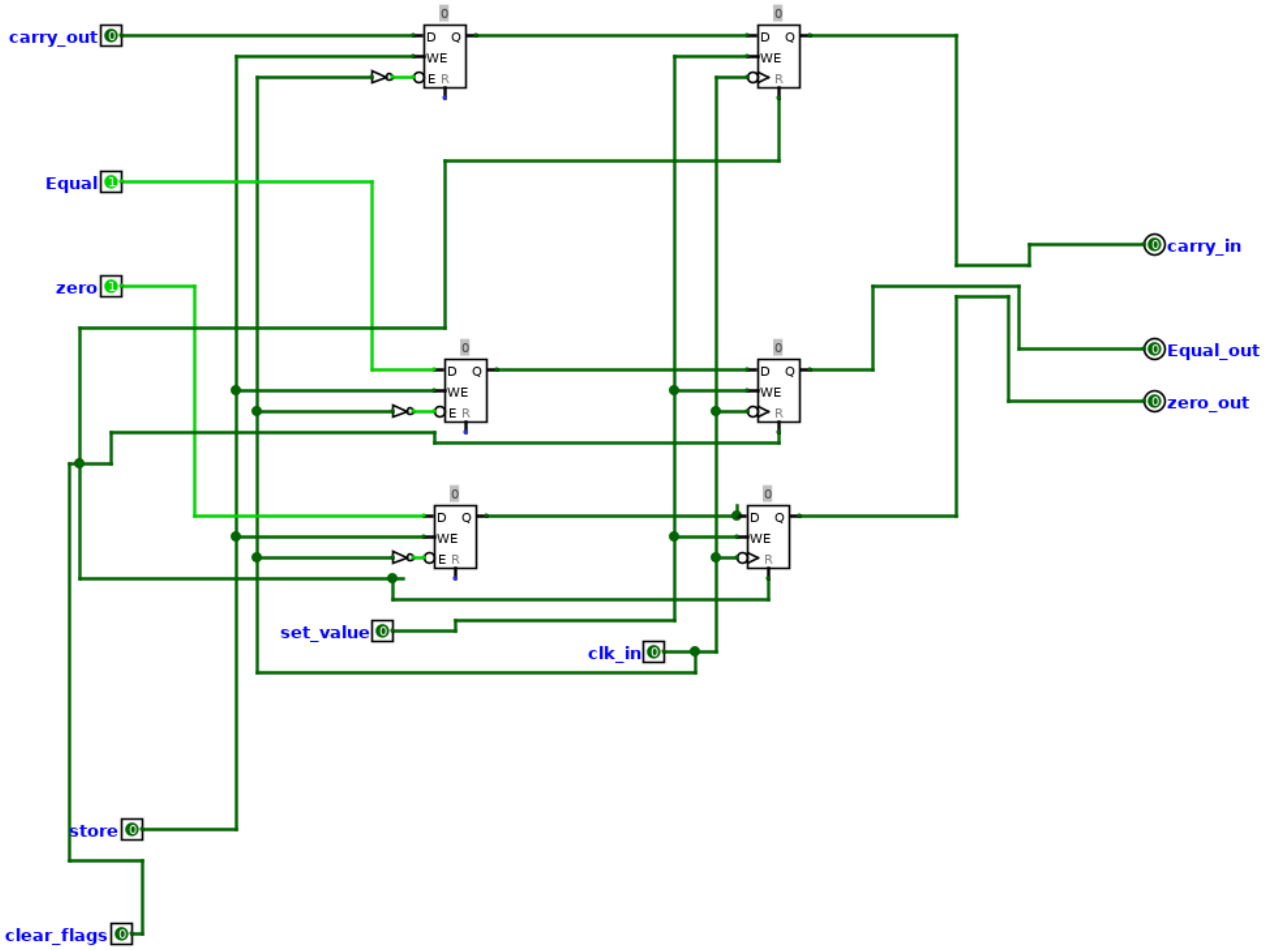




Jednostka Arytmetyczno-Logiczna (*ALU*) wykonuje operacje na dostarczonych do niej wartościach. Operacje realizowane przez ten układ to dodawanie, odejmowanie, XOR, AND, OR. Na podstawie 4 bitów, które z kodu operacji trafiają do *ALU* wybierane jest na multiplekserze, po prawej stronie układu, który z wchodzących do niego sygnałów podać jako wyjście *ALU po operacji.* Dekoder u dołu służy jako zapewnienie, by bit pożyczki/przeniesienia, był generowany tylko, przy operacjach dodawania/usuwania.

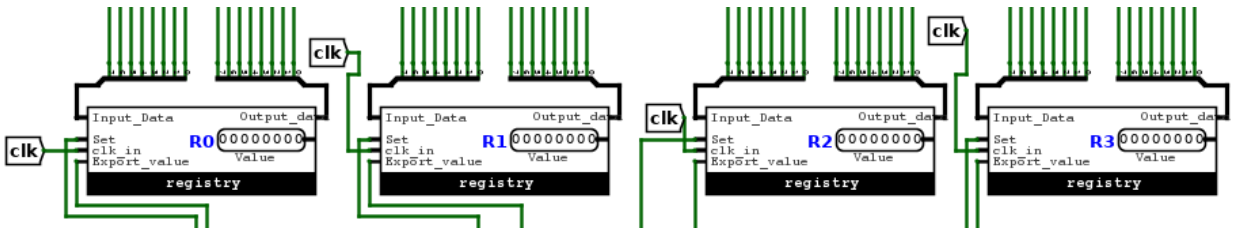
Rejestr flag





Rejestr Flag zapisuje flagi wygenerowane przy operacjach *ALU* oraz dostarcza je do *ALU* do wykorzystania w bieżącej operacji, warto zauważyć, że nie ma osobnej flagi dla przeniesienia i pożyczki, flaga jest dla tych dwóch wartości wspólna. Co oznacza, że jeśli wykonamy dodawanie, które wygeneruje przeniesienie, a następnie wykonamy odejmowanie nie czyszcząc flag, odejmowanie wykona się z pożyczką, mimo iż pożyczka nigdy nie została wygenerowana przez poprzednią operację odejmowania.

Bank Rejestrów

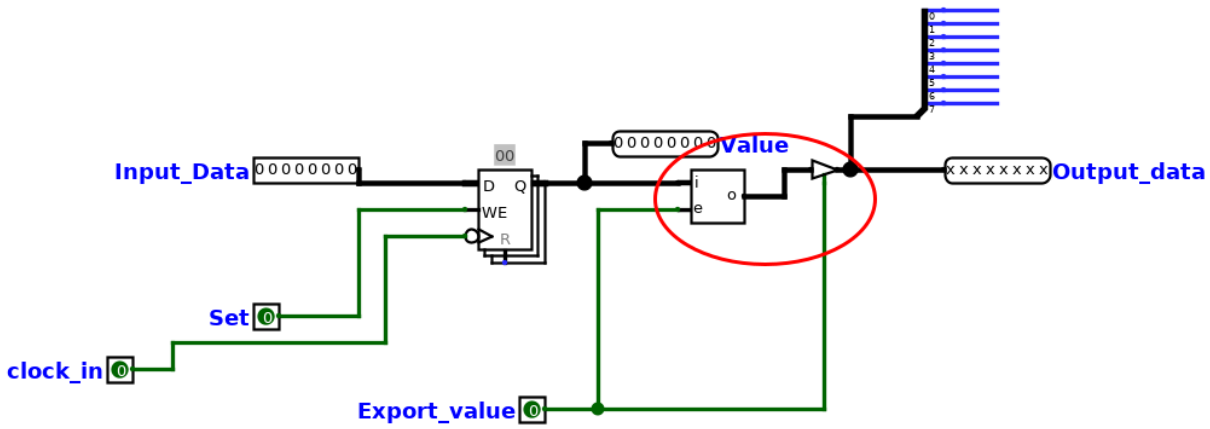


Rejestry, czyli pamięć przechowująca wartości. Bank rejestrów służy programiście, jako miejsce do zapisywania danych, eksportowania ich do używania w różnych operacjach itd.

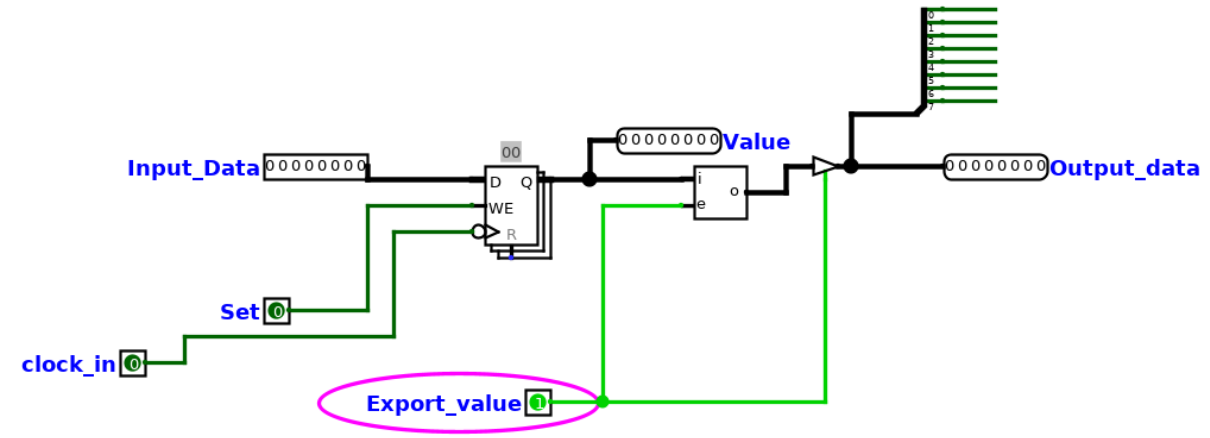
Ważnymi elementami, które posiada każdy komponent tego mikrokontrolera są wejścia sterujące „*Set*” oraz „*Export_value*” zostaną omówione dokładniej w opisie jednostki sterującej. najważniejszą dotyczącą ich informacją jest, że wysoki sygnał „Set” w połączeniu z opadającym zboczem zegara sprawia, że w rejestrze zapisuje się wartość znajdująca się na jego wejściu (*Input_Data*), a wysoki sygnał *Export_value* otwiera bufor znajdujący się na wyjściu rejestru, eksportując jego wartość przez wyjście *Output_data*. *Input* oraz *Output* podłączone są do magistrali służącej do transportu danych.

Magistrala
(na obrazie całego układu zestaw obwodów otaczających wszystkie komponenty)

Magistrala służy do transportowania danych między komponentami procesora, jednocześnie może znajdować się na niej tylko jedna wartość (zestaw sygnałów). Dlatego ważne jest zastosowanie bufora w komponentach pamięci, który w przypadku nie otrzymania sygnału *Export*, nie tylko nie eksportuje wartości na magistralę, ale też „odcina” wyjście rejestru od niej. W przeciwnym wypadku dochodzi do błędów wynikających z kilku zestawów sygnałów trafiających jednocześnie do magistrali (w programie Logisim oznaczone jest to czerwonym kolorem przewodu). Przewód „odcięty” przyjmuje stan nieokreślony oznaczany kolorem niebieskim (Zdj poniżej)

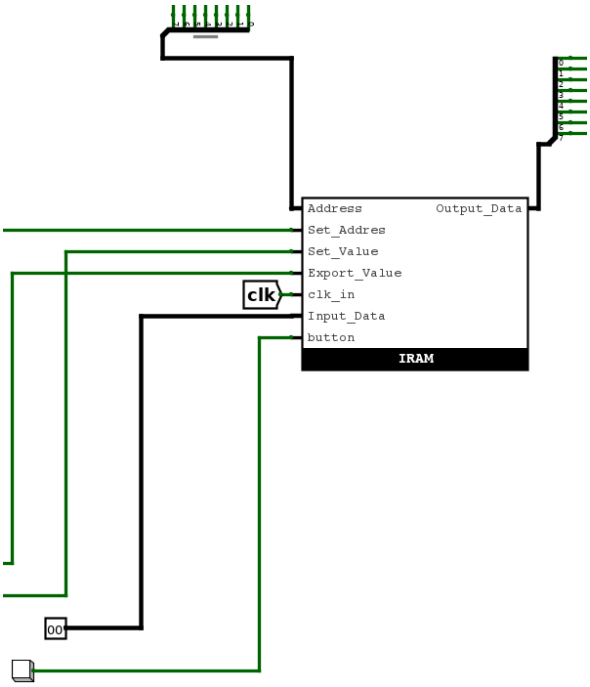


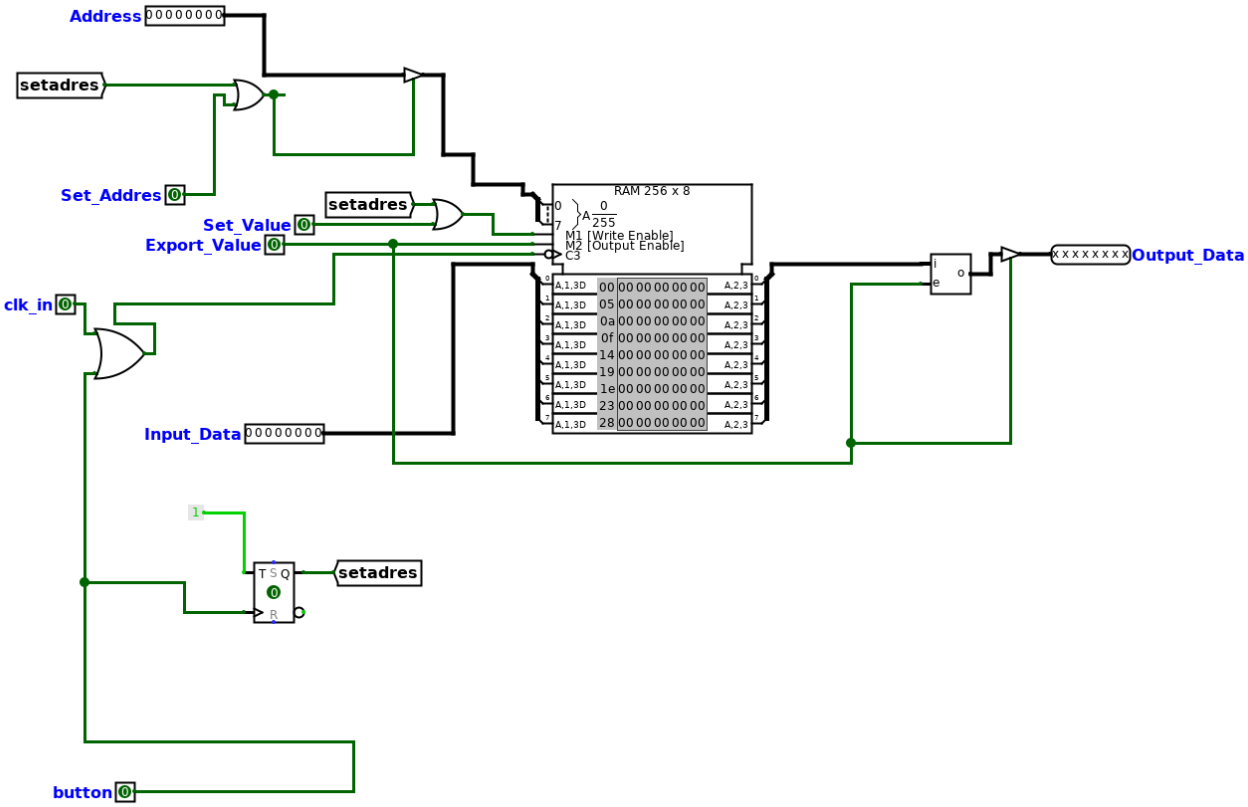
czerwonym okręgiem zaznaczony jest bufor, pin wyjściowy *Output_data* zawiera w sobie „*xxxxxxxx*” oznaczające stan nieokreślony, rozdzielacz znajdujący się za buforem, pokazuje niebieski kolor poszczególnych przewodów.



Wysoki stan *Export_value* sprawia, że stan rejestru zostaje eksportowany do wyjścia.

IRAM

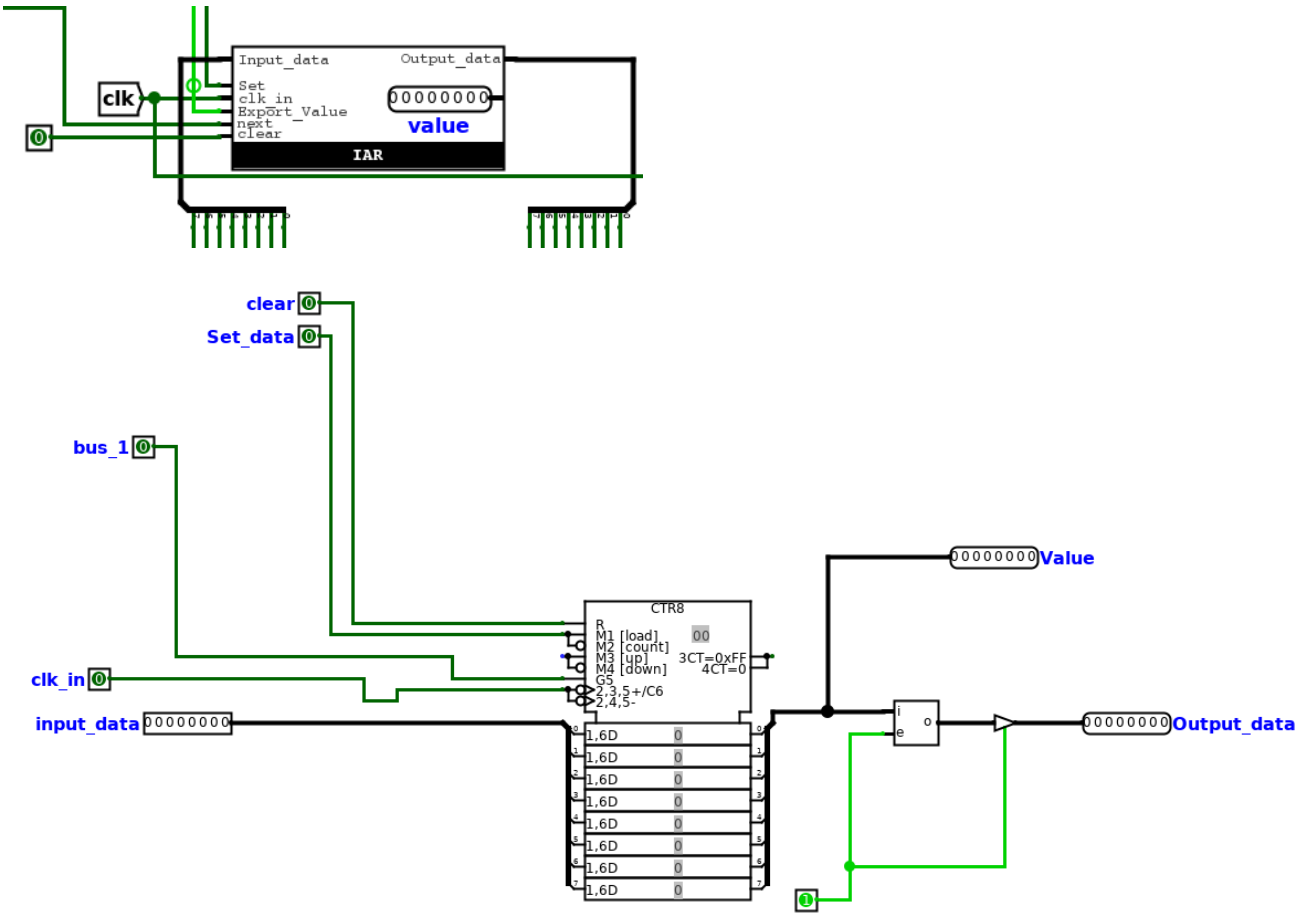




komponent ten służy do przechowywania programu. Wysoki stan „*Set_Address*” w połączeniu z opadającym zboczem zegara, ustawia aktualny adres na wartość znajdującą się na wejściu „*Address*”.
Następnie wysoki stan *Export_value* eksportuje wartość znajdującą się pod aktualnym adresem na wyjście. Wysoki stan pinu Set_value razem z opadającym zboczem zegara, zapisuje wartość

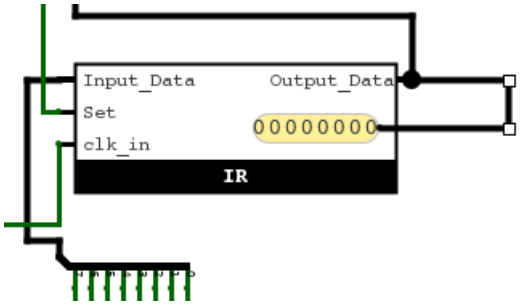
znajdującą się na wejściu, w komórce aktualnego adresu. Główną funkcją komponentu w tym układzie jest załadowanie napisanego wcześniej „programu” czyli kodów operacji na kolejne komórki pamięci a następnie przesyłanie zawartości kolejnych komórek do jednostki sterującej.

Rejestr Adresów Instrukcji



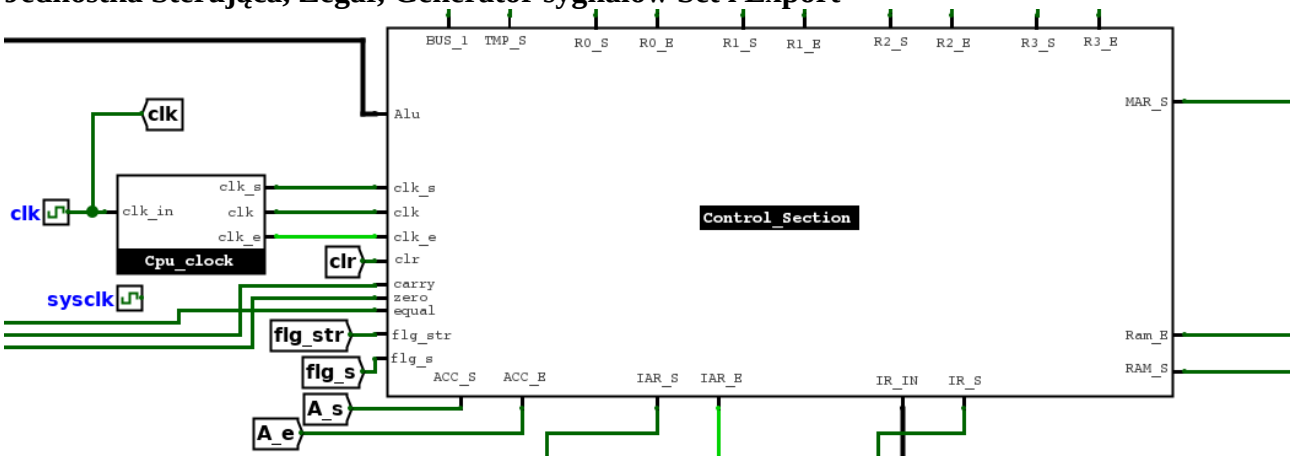
komponent ten – *IAR* dostarcza adresów kolejnych instrukcji, z każdym cyklem procesora zwiększa wartość o 1 dając dostęp do następnego kodu w *IRAM’ie* (dokładne działanie opisane zostanie w opisie Jednostki Sterującej). W przypadku skoków zamiast zwiększyć wartość o 1, ustawiana jest ona na adres do którego wykonywany jest skok (na którym powinna się znajdować kolejna instrukcja)

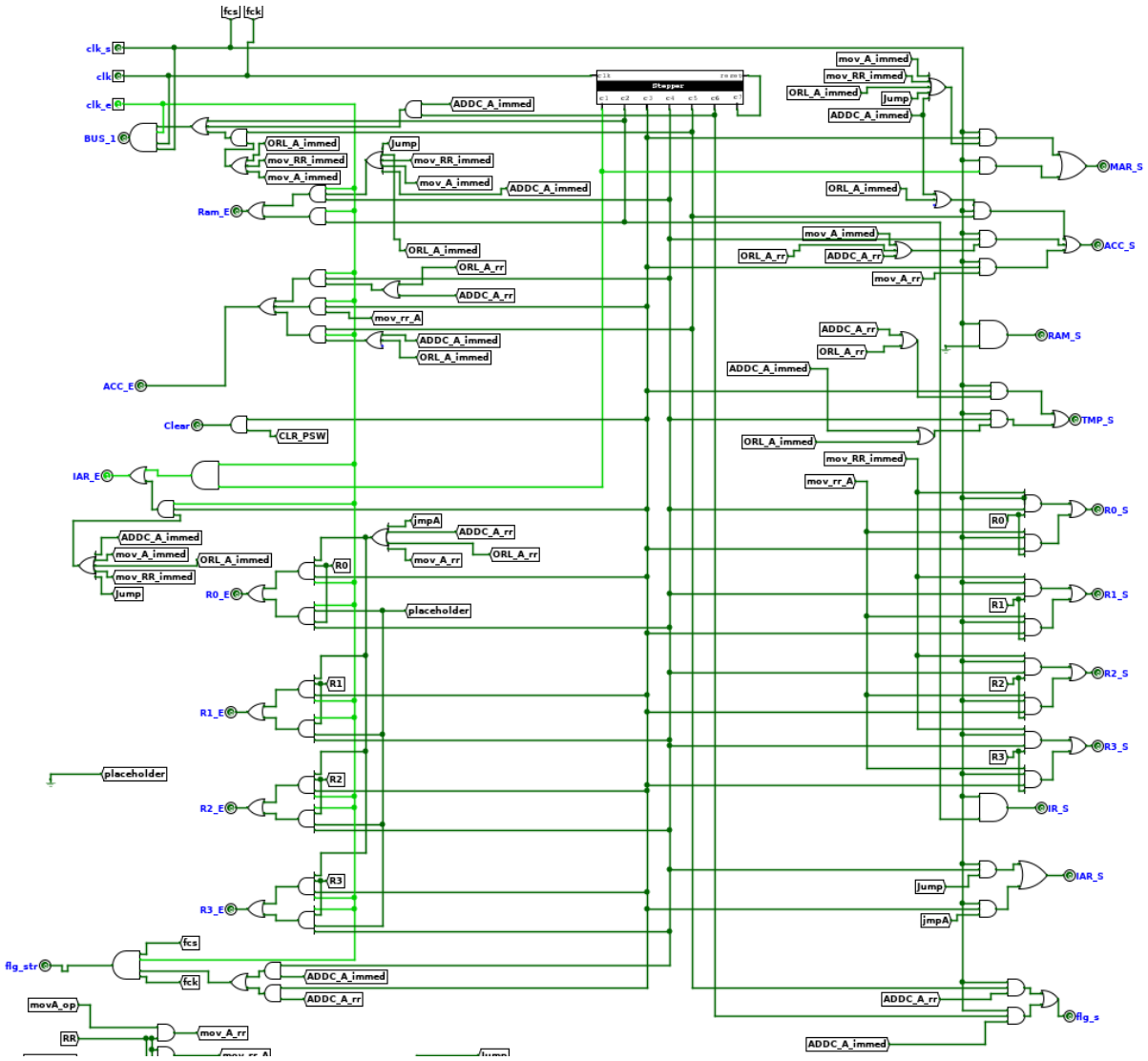
Rejestr Instrukcji

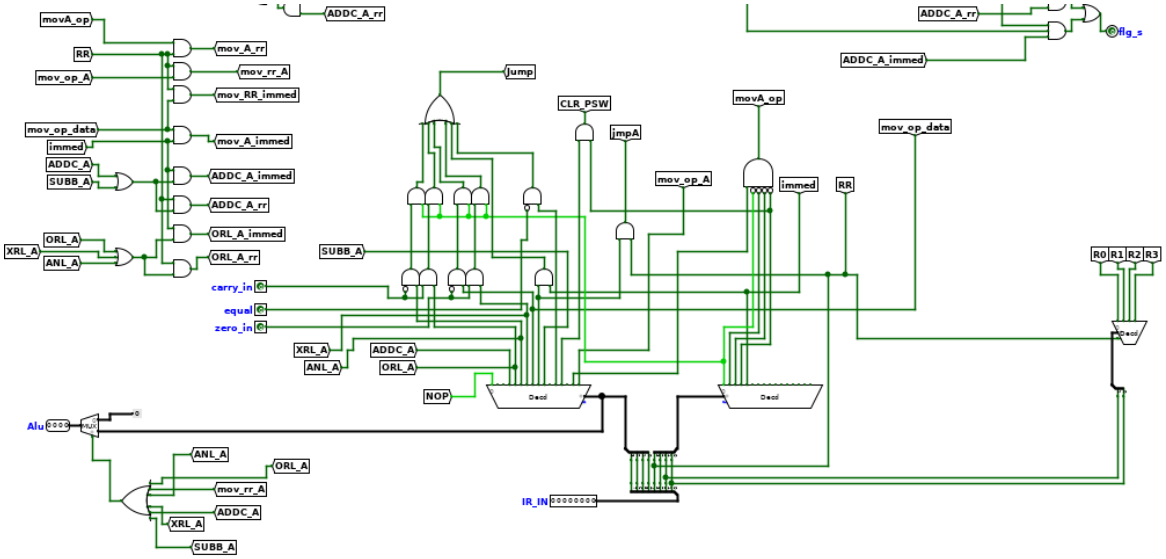


Rejestr do którego zapisywana jest wartość, znajdująca się pod aktualnym adresem w IRAM reprezentująca kod operacji. Rejestr ten dostarcza tę wartość do jednostki sterującej, która na jej podstawie wykonuje operację.

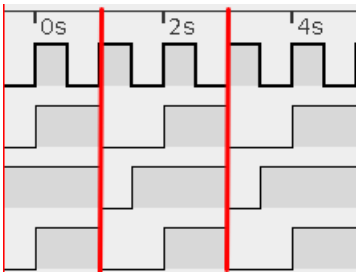
Jednostka Sterująca, Zegar, Generator sygnałów Set i Export







Zegar jest podłączony do komponentu Cpu_clock tam za pomocą przełącznika JK generuje sygnał Export oraz Set na podstawie pulsów zegara głównego:



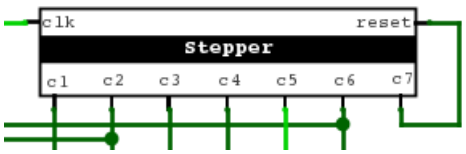
zegar główny

zegar główny podzielony przez 2

sygnał Export

sygnał Set

sygnał zegara trafia do wszystkich komponentów, a jego 2 razy wolniejsza wersja do licznika jednostki sterującej, licznik jest to prosty komponent, który z każdym pulsem zegara podaje sygnał na kolejne ze swoich wyjść, po dojściu do ostatniego, resetuje się i zaczyna od nowa

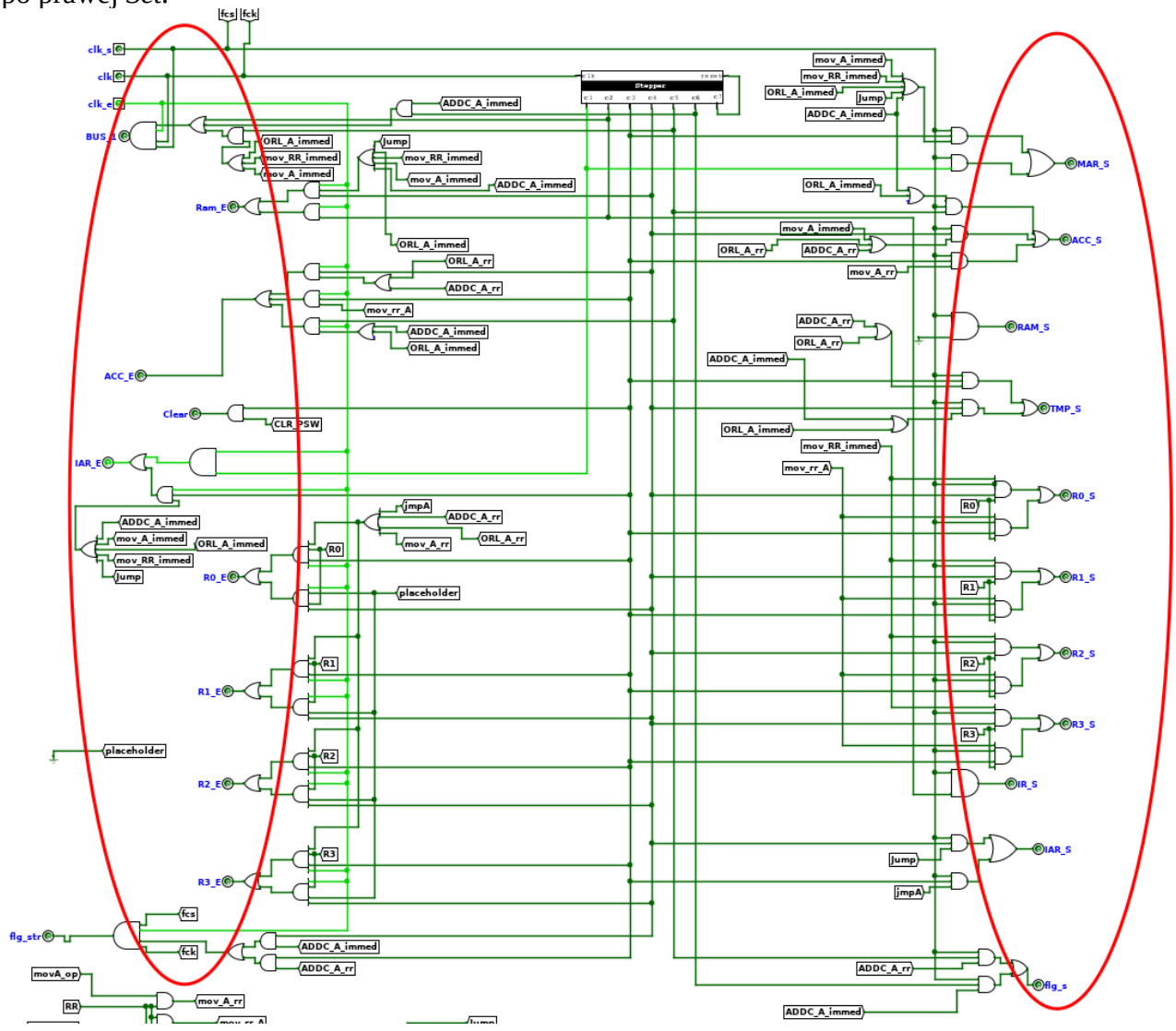


Przejście wszystkich kroków kończy jeden cykl jednostki sterującej. W kolejnych krokach wykonywane są kolejne składowe operacji, aktualnie wykonywanej przez procesor. We wszystkich krokach (z wyjątkiem początkowego stanu procesora gdzie sygnał *Export* = 1), proces wygląda identycznie dla sygnałów *Set* i *Export*:

1. Oba sygnały niskie
2. Sygnał *Export* wysoki
3. Sygnał *Set* wysoki
4. Oba sygnały wysokie
5. Przejście do następnego kroku (i oba sygnały niskie)

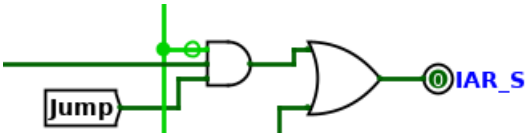
Czerwone linie na wykresie czasowym oddzielają kolejne kroki. Istotne jest, że zawsze najpierw pojawia się sygnał *Export*, następnie *Set*, w trakcie kiedy oba sygnały są wysokie, następuje opadnięcie zbocza głównego zegara, a na końcu oba sygnały opadają przy przejściu do kolejnego kroku.
Jest to związane z działaniem komponentów. Przy otrzymaniu sygnału *Export*, ich zawartość trafia na magistralę. Przy sygnale Set i opadającym zboczu zegara głównego, wartość na ich wejściu (wejścia są podłączone do magistrali) zostaje w nich zapisana. Takie więc następowanie po sobie sygnałów, pozwala na następujący schemat działania: *W każdym kroku możemy wyeksportować zawartość dokładnie jednego komponentu na magistralę, i zapisać ją do jednego lub więcej komponentów.*

Na schemacie okablowania jednostki sterującej po lewej stronie znajdują się piny sygnałów *Export*, po prawej *Set*.



To czy pin sygnałów Export/Set zostanie ustawiony jako „1” można przedstawić w sposób:
jeśli jest to krok X i sygnał clk_e/clk_s jest „1” i sygnał opisujący operację Y jest „1” pin zostanie zasilony.

Wiele operacji „zazębia się” ze sobą, wykonując w niektórych krokach te same działania. Dlatego też widać w wielu miejscach kilka sygnałów wpiętych razem do bramki OR.



Ten prosty przykład pokazuje instrukcję skoku, która zawsze w danym kroku musi wysłać sygnał *Set* do *IAR. P*ionowy zasilony przewód to sygnał clk_s. Więc jeśli po otrzymaniu kodu operacji i „rozkodowaniu” go przez jednostkę sterującą, okaże się że jest to instrukcja skoku. Połączenie opisane Jump zostanie ustawione na „1”. Kiedy trzeci sygnał będący przyłączony do odpowiedniego kroku licznika zostanie zasilony (kiedy licznik dojdzie do tego kroku), zostanie wysłany sygnał *Set* do IAR, który wraz z opadającym zboczem zegara głównego, zapisze w nim wartość podaną mu na wejście.

Piny Set i Export to główne sygnały sterujące tej jednostki i są wykorzystywane niemal przy każdej operacji.

Pierwsze 2 kroki jednostki sterującej nigdy się nie zmieniają i służą pozyskaniu instrukcji z IRAM’u.

Pozyskanie Instrukcji

- W pierwszym kroku, zawartość IRAM eksportowana zostaje na magistralę, oraz wykorzystana przez IRAM aby ustawić go na adresie wskazanym przez tą wartość.
- W drugim kroku, zawartość wybranej wcześniej komórki zostaje eksportowana na magistralę i zapisana w IR , a następnie wartość w IAR zostaje zwiększona o 1.

W ten sposób pozyskaliśmy kod operacji z IRAM i załadowaliśmy go do jednostki sterującej, oraz zapewniliśmy, że następny adres z którego pozyskamy instrukcję na początku następnego cyklu jest następnym, po obecnym, adresem.

Następnie kod operacji zostaje odpowiednio „rozkodowany”, za pomocą bramek logicznych i dekoderów. Na podstawie tego „rozkodowania” ustawiane są sygnały, które sterować będą zachowaniem procesora w następnych krokach cyklu.

4. Spis obsługiwanych instrukcji:

Kod Operacji	Mnemonik
34	ADDC A, #immed
38	ADDC A, R0
39	ADDC A, R1
3A	ADDC A, R2
3B	ADDC A, R3
40	JC
44	ORL A, #immed
48	ORL A, R0
49	ORL A, R1
4A	ORL A, R2
4B	ORL A, R3
50	JNC
54	ANL A, #immed
58	ANL A, R0
59	ANL A, R1
5A	ANL A, R2
5B	ANL A, R3
60	JZ
64	XRL A, #immed
68	XRL A, R0
69	XRL A, R1
6A	XRL A, R2
6B	XRL A, R3
70	JNZ
74	MOV A, #immed
78	MOV R0, A
79	MOV R1, A
7A	MOV R2, A
7B	MOV R3, A
84	JMP
88	JMP R0
89	JMP R1
8A	JMP R2
8B	JMP R3
94	SUBB A, #immed

98	SUBB A, R0
99	SUBB A, R1
9A	SUBB A, R2
9B	SUBB A, R3
C3	CLR PSW (flagi)
B0	CJNE
E8	MOV A, R0
E9	MOV A, R1
EA	MOV A, R2
EB	MOV A, R3
F8	MOV R0, A
F9	MOV R1, A
FA	MOV R2, A
FB	MOV R3, A

W przypadku instrukcji z argumentem będącym bezpośrednią wartością liczbową - „#immed”, należy tą wartość w procesie „kompilowania programu” umieścić pod adresem IRAM następującym zaraz po instrukcji która go wykorzystuje.

dla przykładu „MOV A, #9F” wyglądać będzie następująco:

...	74	9F	...

Dla wszystkich instrukcji skoku oprócz 88-8B należy na tej samej zasadzie, w następnej komórce umieścić adres, do którego zostać ma wykonany skok, a na tym adresie następną intrukcję. Instrukcje 88-8B skaczą do adresu równego zawartości rejestru podanego jako argument.

5. Przykładowy program:

```
MOV A, #AA
ANL A, #F5
MOV R0, A
MOV R1, #0C
ORL A, R1
MOV R1, A
XRL A, #A3
MOV R2, A
ADDC A, #F1
JC #13
MOV R3, A
SUBB A, #00
```

zawartość pliku tekstowego z tym programem, który można załadować do IRAM:

v2.0 raw
74 aa 54 f5 f8 79 c 49
f9 64 a3 fa 34 f1 40 13
0 0 0 fb 94

po zakończeniu programu:

Akumulator: 11111111
R0: 10100000
R1: 10101100
R2: 00001111
R3: 00000000
ustawione flagi carry i equal

(plik z programem załączony do cyfrowej wersji dokumentacji)

6. Statystyki Układu

Component	Library	Simple	Unique	Recursive
TOTAL (with subcircuits)		65	439	544
TOTAL (without project's subcircuits)		51	421	523

7. Wnioski

Zbudowanie układu tego typu jest niezwykle trudnym zadaniem. Ilość informacji, które trzeba przyswoić jest ogromna. Aby układ działał poprawnie należy zadbać o kompatybilność i odpowiednie zgranie ze sobą wielu komponentów jednocześnie. Jeśli mogę pozwolić sobie na odrobinę prywaty, był to najbardziej wyzywający, a jednocześnie satysfakcjonujący i ciekawy projekt dotychczas wykonywany przeze mnie w trakcie studiów. Projektowanie tego mikrokontrolera pozwala w bardzo wyraźny i jasny sposób zobaczyć granicę i „moment przejścia”, pomiędzy programem, a elektroniką. Można dokładnie zaobserwować jak linie kodu stają się impulsami w układzie i jaką rolę i wpływ na poszczególne komponenty te impulsy mają.
Projekt tego rodzaju jest niezwykle obszerny i nawet mimo wielu godzin pracy włożonych w jego realizację, jest to zaledwie część całego chipu 8051. Brakuje wielu elementów i funkcji jak np. DPTR i obsługa wyjątków. Wiele zaimplementowanych elementów nie posiada pełni funkcji oryginalnego 8051. Budzi to szacunek i podziw dla złożoności takiej konstrukcji, która w porównaniu do dzisiejszych produktów komercyjnych jest przecież niezwykle prosta.

8. Bibliografia

[1] oficjalna strona programu Logisim http://www.cburch.com/logisim/

[2] schemat architektury intel 8051 - https://upload.wikimedia.org/wikipedia/commons/thumb/c/cd/Intel_8051_arch.svg/1200px-Intel_8051_arch.svg.png

[3] lista rozkazów - http://www.zsk.ict.pwr.wroc.pl/zsk/repository/dydaktyka/ptm/lab/8051instrset.pdf

[4] Angielska strona wikipedii dotycząca tego mikrokontrolera dostarcza trochę informacji na temat kodowania operacji - https://en.wikipedia.org/wiki/Intel_MCS-51

[5] But How Do It Know? - The Basic Principles of Computers for Everyone by J Clark Scott

9. Repozytorium

https://github.com/Greezye625/OiAK_Projekt_2019