Artur Szymkiewicz
W ramach projektu, zadaniem było opracowanie mikroprogramowalnego układu sterowania, który jest zdolny realizować zestaw określonych rozkazów opisanych w tematach projektów.
Projekt obejmuje trzy główne elementy:

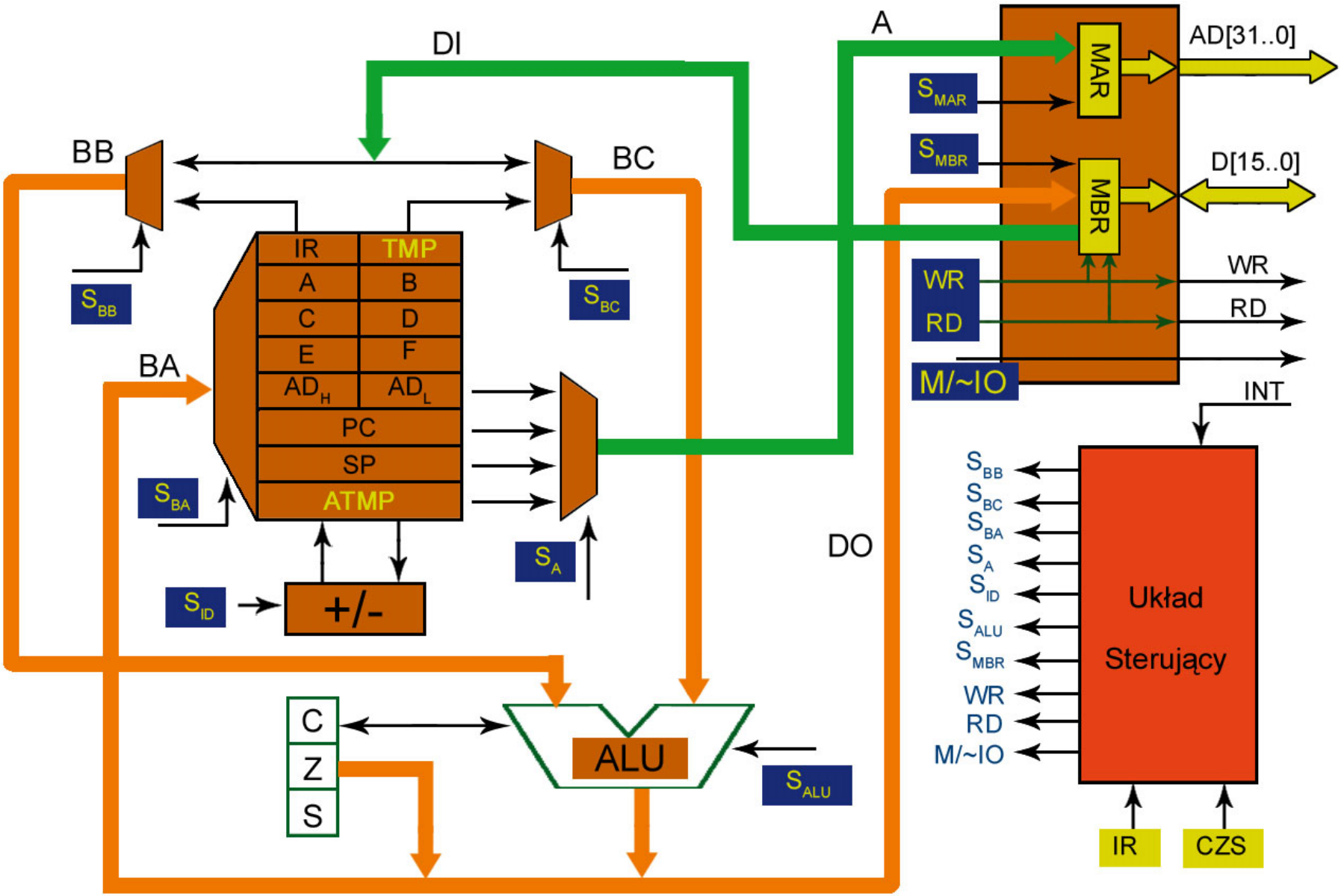
1. **Zaproponowana Struktura Układu:**
    - W tym aspekcie projektu należało określić fizyczną organizację układu, jego bloków funkcjonalnych, połączeń, oraz interfejsów. Struktura układu powinna być zoptymalizowana pod kątem efektywnego wykonania rozkazów, minimalizacji zasobów, oraz spełnienia wymagań projektu.
2. **Mikroprogram w Formie Listy:**
    - Mikroprogram to sekwencja mikrooperacji, które są wykonywane w odpowiedzi na odczytane instrukcje. Mikroprogram został napisany w postaci listy, co oznacza, że dla każdej instrukcji rozkazu określono kroki wykonywane przez układ w postaci operacji mikroprogramu. Mikrooperacje te obejmują kontrolę dostępu do pamięci, operacje arytmetyczne, logiczne, oraz inne manipulacje rejestrami.
3. **Zakodowana Wersja Mikroprogramu:**
    - Zakodowana wersja mikroprogramu to efekt przekształcenia listy mikrooperacji na formę, która jest zrozumiała i wykonywalna przez układ cyfrowy. Obejmuje to przypisanie konkretnych kodów operacyjnych do poszczególnych mikrooperacji, tak aby układ mógł poprawnie interpretować i wykonywać zadane operacje.

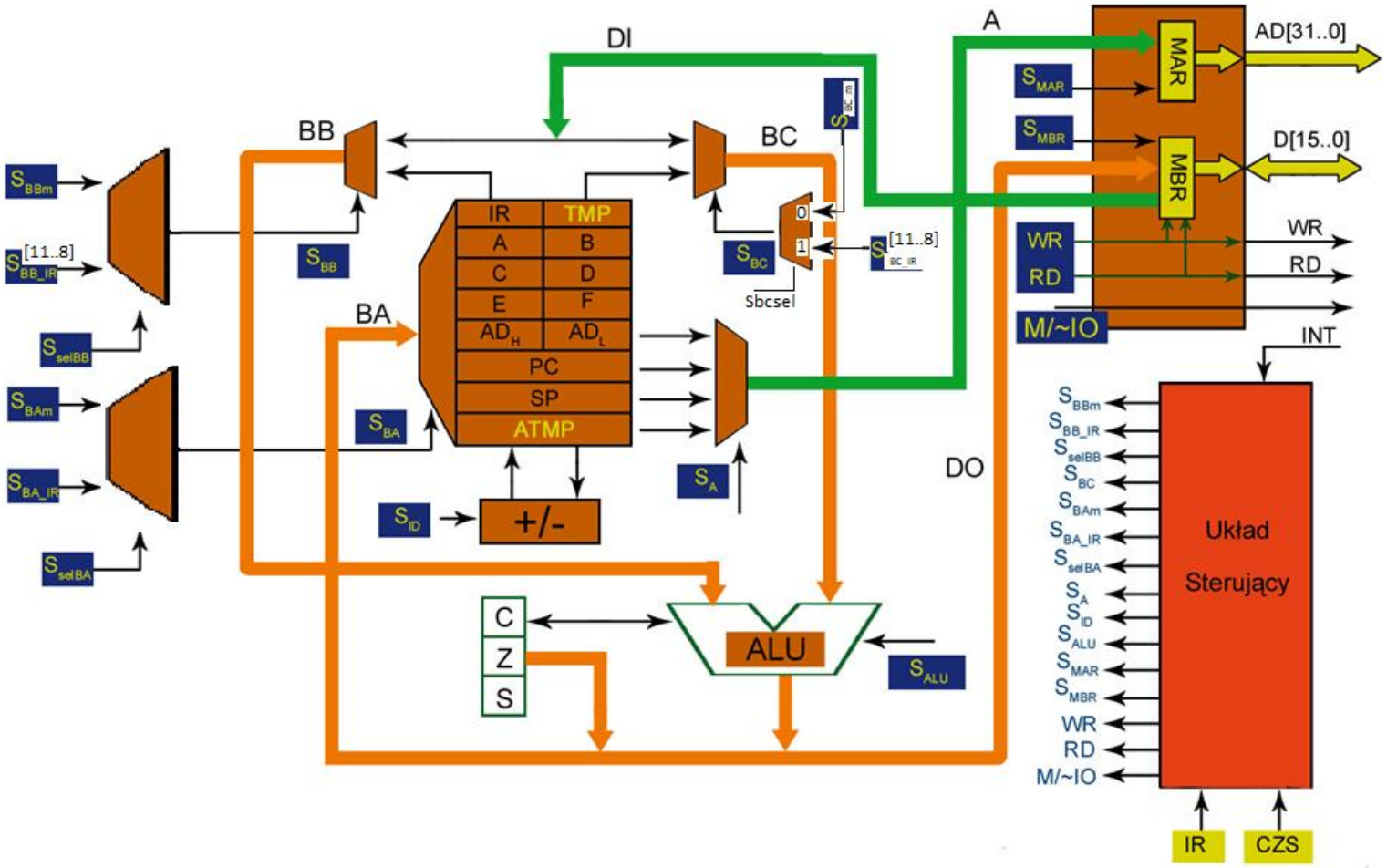
Oznaczenia przyjęte w opisie listy rozkazów procesora:
REG - rejestr 16-bitowy A, B, C, D, E, F, ADH, ADL,
RA - rejestr 32-bitowy AD lub SP,
[RA] - adres argumentu w rejestrze (adresowanie rejestrowe pośrednie,
st16[RA] – adresowanie bazowe,
adr - adres 32-bitowy (adresowanie bezpośrednie),
[adr] - adresowanie pośrednie,
st8 - stałą 8-bitowa,
st16 - stała 16-bitowa,
st32 - stała 32-bitowa.

Zakładam, że pierwsze słowo każdego rozkazu ma następującą budowę:
B15..B8 - numer grupy rozkazów
B2..B0 – numer rozkazu w grupie

LISTA ROZKAZÓW :
1. **NOT REG** negacja rejestru roboczego
2. **VINC [adr], n** inkrementacja wektora w pamięci
3. **XCHG REG, [RA]** wymiana danych rejestr-pamięć

Propozycja struktury układu przedstawiona na rysunku 1 stanowi fundament mojego projektu. Jednakże, aby zwiększyć elastyczność i zoptymalizować wydajność, wprowadziłem pewne modyfikacje, co obrazuje rysunek 2.
Na rysunku tym uwzględniłem dodanie multiplekserów do szyn sterowania S_BA, S_BB i S_BC. Te modyfikacje umożliwiają precyzyjne sterowanie przepływem danych do i z 16-bitowych rejestrów procesora. Decyzje dotyczące źródła i przeznaczenia danych są teraz bezpośrednio pobierane z pól adresowych aktualnej instrukcji assemblerowej, przechowywanej w rejestrze IR. Oparte na tym podejście $S_{Bc-IR} = IR[15..12]$
Wprowadzenie tych modyfikacji przynosi znaczne korzyści w procesie mikroprogramowania. Można teraz napisać jedną mikroprocedurę dla każdego rodzaju rozkazu. W przeciwnym przypadku, gdybym trzymał się klasycznej struktury układu, musiałbym opracować 16 mikroprocedur dla operacji korzystających z rejestru 16-bitowego do odczytu lub zapisu, a także 256 mikroprocedur dla operacji korzystających z rejestru 16-bitowego do jednoczesnego odczytu i zapisu. Ta modyfikacja ma zatem istotne znaczenie dla projektanta, umożliwiając bardziej zwięzłe i zrozumiałe mikroprogramowanie.
Choć rozważałem również dodanie multipleksera na wyjściu rejestrów 32-bitowych, zdecydowałem się na napisanie oddzielnych mikroprocedur dla rejestrów AD i SP, biorąc pod uwagę, że aktualnie korzystam tylko z dwóch z tych rejestrów, a korzyść z tego rozwiązania byłaby ograniczona.
W projekcie przyjąłem, że układ stosuje big-endian - młodszy bajt (najmniej znaczący) jest przechowywany na końcu, a starszy bajt (najbardziej znaczący) na początku.


DI
A
BB
BC
BA
DO
IR
TMP
A
B
C
D
E
F
ADH
ADL
PC
SP
ATMP
+/-
SBB
SBC
SBA
SID
SA
SMAR
SMBR
MAR
MBR
AD[31..0]
D[15..0]
WR
RD
M/~IO
INT
C
Z
S
ALU
SALU
Układ
Sterujący
IR
CZS


DI
A
BB
BC
BA
DO
IR
TMP
A
B
C
D
E
F
AD_H
AD_L
PC
SP
ATMP
+/-
ALU
C
Z
S
MAR
MBR
AD[31..0]
D[15..0]
WR
RD
M/~IO
INT
Układ
Sterujący
IR
CZS
Sbcsel

2 . MIKROPROGRAM ZAPISANY W POSTACI LISTY

1. Cykl pobierania rozkazu

Cykl pobierania rozkazu polega na odczytaniu z pamięci zawartości komórki, której adres znajduje się aktualnie w rejestrze PC i zapisaniu danych do rejestru IR

adres 0000 :
A <= PC (S_A=1); MAR <= A (S_{MAR} =1); MBR1 <= D (S_{RD}=1); BB <= DI <= MBR1 (S_{BB}=S_{BBm}= 0, S_{selBB}= 0); BA <= BB (S_{ALU}=0);
IR <= BA (S_{BA}=S_{BAm}= 0,S_{selBA}= 0);

1. Cykl dekodowania rozkazu

adres 0001 :
PC <= PC + 1 (S_{ID}= 1) goto IR[2..0] * 4;

adres 0004:

```
REG => BB (SBB=1,IR[15..12],SBB_sel=1,SBBm=1);// Przygotowanie rejestru BB do
operacji NOT
BB => ALU(S_ALU=8), //Wykorzystanie ALU do ustawienia flagi N na negację zawartości BB
ALU => BA (S_BA=1, S_BAm=1, S_selBA=1),// Kopia BA
ALU => DO => MBR => D[15..0] // Musimy coś zapisać
BA => REG  // Aktualizacja rejestru REG
KONIEC → 0000/goto 0;
```

```
adres 0008:
PC->MAR->AD; D->MBR->BB[SBb=1, IR[15..12],Sbb_sel=1, Sbbm=1]->ALU=>BA=>ATMPh,
RA++,goto 2;
adres 0002:
PC->MAR->AD;D->MBR=>BB->ALU=>BA[Sba=1, IR[15..12],Sba_sel=1, Sbam=1]-
>ATMPl,RA++,PC++;
Adres 0003:
ATMP->MAR->AD; D->MBR->BB[SBb=1, IR[15..12],Sbb_sel=1, Sbbm=1]->ALU=>BA=>TMP,
RA++,goto 5;
Adres 0005:
ATMP->MAR->AD; D->MBR->BB[SBb=1, IR[15..12],Sbb_sel=1, Sbbm=1]->ALU=>BA=>ATMPl, PC++;
Adres 0006:
TMP->MAR->AD; D->MBR->BB[SBb=1, IR[15..12],Sbb_sel=1, Sbbm=1]->ALU=>BA=>ATMPh, PC++,
RA++;
Adres 0007:
A=>MAR->AD; D->MBR->BB[SBb=1, IR[15..12],Sbb_sel=1, Sbbm=1]-
>ALU=>BA=>A;ALU=>DO=>MBR=>D;SP=>A=>MAR=>AD;goto 9;
Adres 0009:
PC=>MAR=>AD; RD=1, D=>MBR=>DI=>BB=>ALU=>BA=>TMP,RA++,PC++;
Adres 0010:
ATMP=>A=>MAR=>AD;A=>BB[SBb=1, IR[15..12],Sbb_sel=1,
Sbbm=1]=>ALU[Salu=12]=>BA=>A;ALU=>DO=>MBR=>D, PC++;
Adres 0011:
TMP=> BB[SBb=1, IR[15..12],Sbb_sel=1, Sbbm=1]=>ALU(dec)=>BA=>TMP;if(c==1){goto
15},goto 13,RA++;
Adres 0013:
Goto 10;TMP=> BB[SBb=1, IR[15..12],Sbb_sel=1, Sbbm=1]=>ALU=>BA=>TMP;
```

```
Adres 0014:
SP=>A=>MAR=>D;D=>MBR=>DI=>BB[Sbb=1, IR[15..12],Sbb_sel=1, Sbbm=1]=>ALU=>BA=>A;goto 0;


Adres 0012
RA=>MAR=>AD; D=>MBR=>DI=>BB[Sbb=1, IR[15..12],Sbb_sel=1, Sbbm=1]=>ALU=>BA=>TMP;RA++,
goto 15;
Adres 0015:
REG=>BB[SBb=1, IR[15..12],Sbb_sel=1, Sbbm=1]=>ALU=>BA=>DO=>MBR=>D;RA=>MAR=>AD,goto
17.RA++;
Adres 0017:
TMP=>BB[SBb=1, IR[15..12],Sbb_sel=1, Sbbm=1]=>ALU=>BA=>REG, PC++, RA++;
Adres 0018:
REG=>Sa=>A=>MAR=>AD;RA=>MBR=>D,PC++,RA++;
Adres 0019:
ATMP=> BB[SBb=1, IR[15..12],Sbb_sel=1, Sbbm=1]=>ALU=>BA=>REG, RA++;goto 0;
```