Układy mikroprogramowane

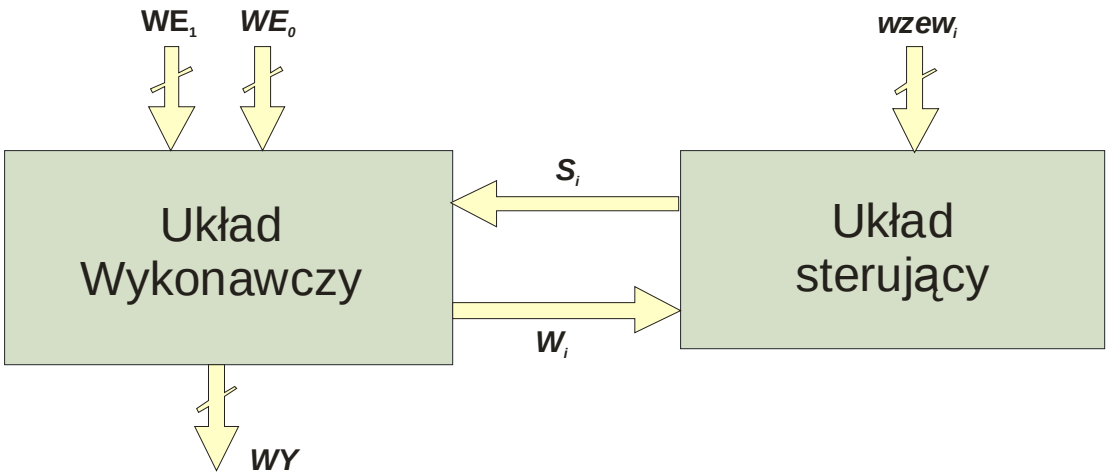
Warszawa, 01/2009

1. Wprowadzenie do mikroprogramowania

Mikroprogramowane układy cyfrowe składają się z dwóch części:

- części wykonującej operacje na danych wejściowych (układ wykonawczy),
- części sterującej wykonywaniem tych operacji (układ sterujący).

Schemat takiego układu przedstawia rysunek 1. Układ sterujący kontroluje pracę układu wykonawczego poprzez linie sterowania S_i. Liniami tymi przesyłane są sygnały wybierające rodzaj operacji, która ma być wykonana przez układ wykonawczy oraz sygnały wskazujące, skąd mają być pobrane argumenty dla tej operacji. Układ wykonawczy informuje o swoim stanie za pomocą sygnałów przesyłanych liniami warunków W_i. Pojawiają się tam np. sygnały mówiące o wyniku ostatniej operacji. Sygnały te wpływają na dalsze działanie jednostki sterującej. Laboratoryjny układ wykonawczy posiada dwa ośmiobitowe wejścia danych **WE$_1$** i **WE$_0$** i szesnastobitowe wyjście danych **WY** (utworzone z bitów zapisanych w dwóch ośmiobitowych rejestrach **RW** i **MK**).



Rysunek 1. Schemat blokowy układu mikroprogramowanego.

Układ sterujący można zaprojektować na dwa różne sposoby: jako automat synchroniczny lub jako układ z pamięcią. Pierwszy sposób polega na zaprojektowaniu takiego automatu synchronicznego, którego działanie (przejścia ze stanu do stanu) jest określone przez algorytm wykonywany przez układ wykonawczy. Na podstawie algorytmu buduje się tabelę przejść i wyjść automatu. Jednym z rozwiązań może być automat zwany rozdzielaczem sterującym, w którym stany kodowane są w kodzie „1 z n” (przerzutników jest tyle, ile jest stanów).

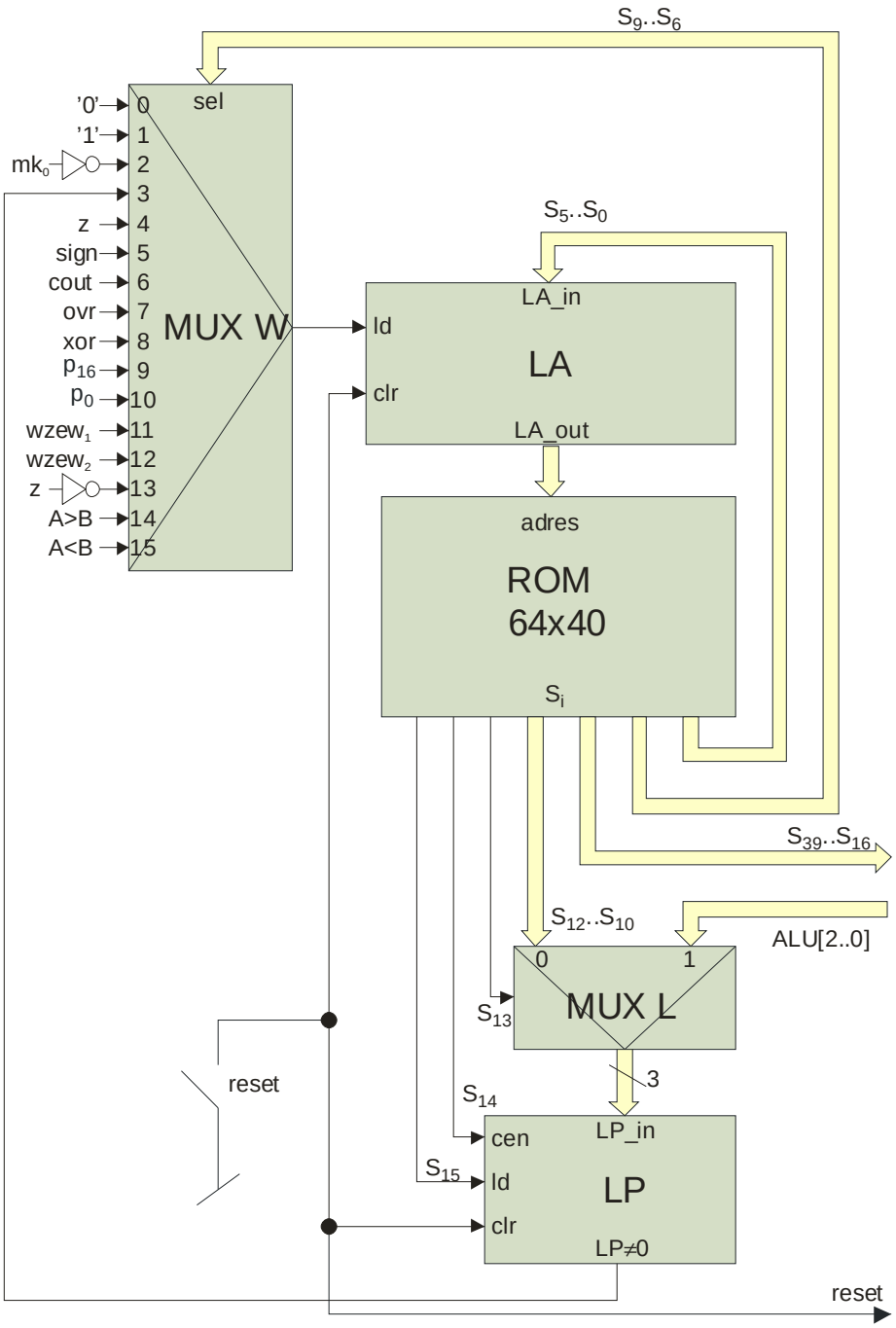
Drugi sposób polega na utworzeniu odpowiedniego programu działającego zgodnie z wymaganym algorytmem i zapisaniu go w pamięci. Układ taki nazywany jest mikroprogramowanym układem sterującym. Kolejne rozkazy programu nazywane są mikrorozkazami lub mikroinstrukcjami. Mikroinstrukcje składają się z mikrooperacji, czyli zestawu bitów przeznaczonych do sterowania poszczególnymi blokami układu. Układ mikroprogramowany wyróżnia się prostotą i przejrzystością budowy. Zastosowanie mikroprogramowania pozwala na łatwą modyfikację działania jednostki sterującej poprzez wymianę mikroprogramu, bez konieczności modyfikowania struktury sprzętowej układu.

2. Przykładowy układ mikroprogramowany

2.1. Układ sterujący

W skład układu sterującego wchodzą:

- pamięć **ROM**,
- układ adresowania (licznik mikrorozkazów **LA** oraz multiplekser skoku **MUX W**),
- układ implementacji pętli (licznik pętli **LP** oraz multiplekser ładowania licznika **MUX L**).



Rysunek 2. Schemat blokowy układu sterującego

Pamięć ROM

Pamięć **ROM** ma pojemność 64 słów 40 bitowych. Bity $S_{39} - S_{16}$ są przeznaczone do sterowania układem wykonawczym. Pozostałe bity $S_{15} - S_0$ odpowiadają za sterowanie układem sterującym. Wybór komórki pamięci następuje poprzez podanie (z układu adresowania) adresu na wejście ***adres*** pamięci **ROM**. W układzie laboratoryjnym moduł pamięci **ROM** zastał zrealizowany jako pamięć RAM, która może być w czasie ćwiczenia zapisywana z komputera poprzez łącze szeregowe RS232.

Układ adresowania

Licznik adresu LA jest 6-bitowym licznikiem, który wykonuje dwie operacje: inkrementację zawartości oraz ładowanie licznika wartością z wyjścia pamięci **ROM**. Wyjście licznika jest podane na wejście ***adres*** pamięci **ROM** i służy do wyboru kolejnego mikrorozkazu.

Gdy na wejściu ***ld*** licznika **LA** jest zero, to następuje inkrementacja jego zawartości, natomiast gdy na wejściu ***ld*** jest jedynka, to licznik zostaje załadowany wartością podaną na wejście ***LA_in*** (czyli adresem skoku zapisanym w mikroprogramie na bitach $S_5 - S_0$). Załadowanie nowej zawartości powoduje wykonanie skoku w mikroprogramie, który może być warunkowy lub bezwarunkowy. Zależy to od stanu na wyjściu **multipleksera warunku MUX W**. Wartość na wyjściu tego multipleksera zależy od jego wysterowania, czyli od wartości bitów $S_9 - S_6$ słowa mikrorozkazu. Jeśli bity te mają wartość 0000, to na wyjściu multipleksera jest zero i jako następny wykonywany jest mikrorozkaz umieszczony pod kolejnym adresem w pamięci **ROM** (inkrementacja licznika **LA**). Jeśli bity te mają wartość 0001, to na wyjściu multipleksera skoku jest jedynka i wykonany zostanie skok bezwarunkowy (ładowanie licznika **LA**). Jeśli bity te mają wartości od 0010 do 1111, to wartość na wyjściu multipleksera zależy od stanu na odpowiednim wejściu, czyli od wybranego warunku skoku. Na przykład, gdy bity $S_9 - S_6$ słowa mikrorozkazu mają wartość 0010, to warunkiem jest zanegowana wartość bitu mk_0. W tabeli 1 przedstawiono możliwości wyboru warunku skoku, a w tabeli 2 sygnały sterujące licznikiem adresu **LA**.

Tabela 1: Sterowanie multiplekserem warunków skoku.

Sterowanie Bity $S_9 - S_6$	*Wejście multipleksera*	*Opis*
0000	0	Inkrementacja licznika adresów
0001	1	Skok bezwarunkowy
0010	$\sim mk_0$	Negacja najmniej znaczącego bitu rejestru **MK**
0011	***LP≠0***	Wyjście ***LP≠0*** licznika pętli **LP**
0100	***z***	Bit zero z rejestru znaczników
0101	***sign***	Bit znaku z rejestru znaczników
0110	***cout***	Bit przeniesienia z rejestru znaczników
0111	***ovr***	Bit nadmiaru z rejestru znaczników
1000	***xor***	Bit będący wartością *XOR* najbardziej znaczących bitów argumentów **ALU**
1001	P_{16}	Bit rw_7 wysuwany z rejestru **RW** podczas przesuwania w lewo
1010	P_0	Bit mk_0 rejestru **MK** podczas przesuwania w prawo

Sterowanie Bity $S_9 - S_6$	*Wejście multipleksera*	*Opis*
1011	***wzew*$_1$**	Warunek zewnętrzny z pakietu z układem FPGA z przełącznika SW3. Gdy ustawiony w pozycji górnej, to na wejście multipleksera podawana jest wartość 0. Gdy ustawiony w pozycji dolnej, to na wejście multipleksera podawana jest wartość 1.
1100	***wzew*$_2$**	Warunek zewnętrzny z pakietu z układem FPGA odczytywany ze złącza SV8 (bit D7)
1101	***~z***	Negacja bitu zero z rejestru znaczników
1110	***A>B***	Argument ***A*** w kodzie NKB większy od argumentu ***B***
1111	***A<B***	Argument ***A*** w kodzie NKB mniejszy od argumentu ***B***

*Tabela 2: Sterowanie licznika adresów **LA**:*

ld	***clr***	*Opis*
1	1	Ładowanie licznika **LA** wartością podaną na wejście ***LA_in*** (bity $S_5 - S_0$)
0	1	Inkrementacja zawartości licznika
X	0	asynchroniczne zerowanie licznika

Licznik pętli LP

Licznik **LP** służy do implementacji w mikroprogramie pętli. Jest to 3-bitowy licznik z asynchronicznym zerowaniem sygnałem ***clr***, zezwoleniem na zliczanie ***cen***, oraz z synchronicznym ładowaniem ***ld***. Wyjście ***LP≠0*** przyjmuje wartość zero, gdy wszystkie bity licznika mają wartość zero, w przeciwnym razie na wyjściu ***LP≠0*** jest jedynka.

Gdy wejście ***ld*** licznika ma wartość jeden, to do licznika wpisywana jest wartość z wyjścia multipleksera **MUX L**. Gdy bit s_{13} ma wartość zero, to na wyjściu multipleksera jest wartość odpowiadająca bitom $S_{12} - S_{10}$ mikrorozkazu. Gdy bit s_{13} ma wartość jeden, to na jego wyjściu jest wartość, odpowiadająca trzem najmłodszym bitom wyjścia układu **ALU**. W tabeli 3 pokazano sposób sterowania licznikiem pętli.

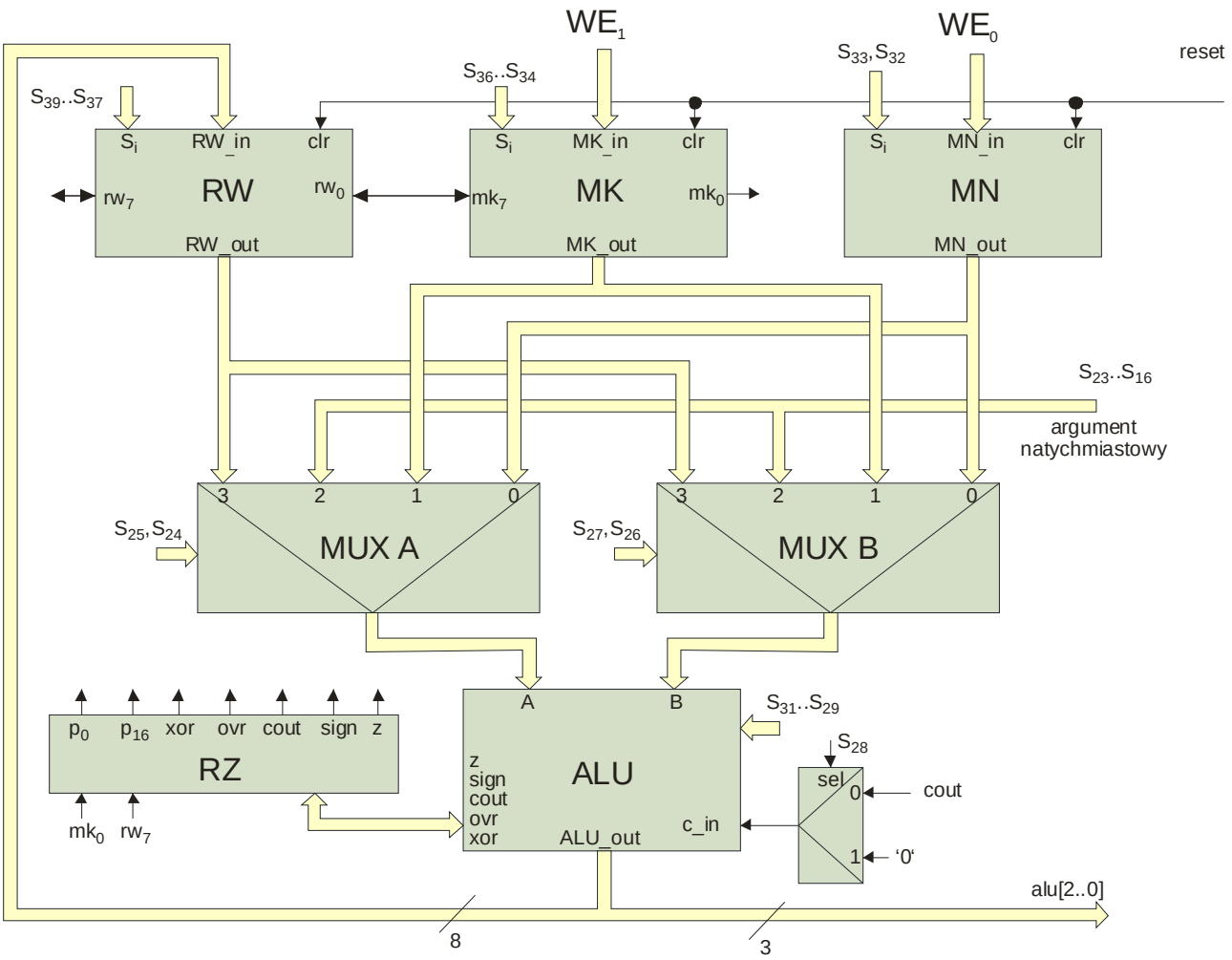
*Tabela 3: Sterowanie licznikiem pętli **LP**:*

Wejścia			*Wyjścia*	*Opis*
clr	***ld*** S_{15}	***cen*** S_{14}	**LP≠0**	
1	X	X	0	Asynchroniczne zerowanie
0	0	0	**LP**≠0	Podtrzymanie zawartości licznika
0	1	x	**LP**≠0	Synchroniczne ładowanie zawartości
0	0	1	**LP**≠0	Dekrementacja zawartości licznika

2.2. Układ wykonawczy

W skład układu wykonawczego wchodzą:

- trzy ośmiobitowe rejestry: **RW** (rejestr wyniku), **MK** (rejestr mnożnika) i **MN** (rejestr mnożnej),
- dwa multipleksery **MUX A** i **MUX B** wyboru argumentów dla układu **ALU**,
- jednostka arytmetyczno-logiczna **ALU**,
- rejestr znaczników **RZ**.



Rysunek 3. Schemat blokowy układu wykonawczego

Rejestry

Rejestr wyniku RW jest 8-bitowym rejestrem przesuwającym. Posiada wejście asynchronicznego zerowania ***clr*** i wejścia sterowania, na które podawane są trzy bity mikrorozkazu $S_{39} - S_{37}$. Rejestr posiada 8-bitowe wejście i 8-bitowe wyjście danych. W czasie przesuwania w lewo na wyjściu rw_7 pojawia się najbardziej znaczący bit rejestru, natomiast podczas przesuwania w prawo najmniej znaczący bit rejestru pojawia się na wyjściu rw_0. W tabeli 4 pokazano sposób sterowania rejestrem **RW**.

Tabela 4: Sterowanie rejestrem RW.

Sterowanie $S_{39} - S_{37}$	*Wejście* ***clr***	*Wyjście* ***RW_out***	*Opis*
X	0	00000000	Zerowanie asynchroniczne
000	1	**RW_out**	NOP
001	1	00000000	Zerowanie synchroniczne
010	1	**RW_in**	Ładowanie
011	1	**RW_out [6..0]** & 0	Przesunięcie w lewo z wsuwaniem 0
100	1	**RW_out [6..0]** & 1	Przesuwanie w lewo z wsuwaniem 1
101	1	**RW_out [6..0]** & mk_7	Przesuwanie w lewo z wsuwaniem bitu mk_7
110	1	0 & **RW_out [7..1]**	Przesuwanie w prawo z wsuwaniem 0
111	1	**cout** & **RW_out [7..1]**	Przesuwanie w prawo z wsuwaniem bitu ***cout***

Rejestr mnożnika MK jest 8-bitowym rejestrem przesuwającym. Rejestr **MK** posiada wejście asynchronicznego zerowania ***clr***. W tabeli 5 pokazano wysterowanie układu bitami $S_{36} - S_{34}$ słowa mikrorozkazu umożliwiające wykonanie odpowiednich operacji.

Tabela 5: Sterowanie rejestrem MK.

Sterowanie $S_{36} - S_{34}$	*Wejście* ***clr***	*Wyjście* ***MK_out***	*Opis*
X	0	00000000	Zerowanie asynchroniczne
000	1	**MK_out**	NOP
001	1	00000000	Zerowanie synchroniczne
010	1	**MK_in**	Ładowanie
011	1	**MK_out[6..0]** & 0	Przesuwanie w lewo z wsuwaniem 0
100	1	**MK_out[6..0]** & 1	Przesuwanie w lewo z wsuwaniem 1
101	1	0 & **MK_out[7..1]**	Przesuwanie w prawo z wsuwaniem 0
110	1	1 & **MK_out [7..1]**	Przesuwanie w prawo z wsuwaniem 1
111	1	rw_0 & **MK_out [7..1]**	Przesuwanie w prawo z wsuwaniem bitu rw_0

Rejestr mnożnej MN jest 8-bitowym rejestrem z asynchronicznym zerowaniem, sterowanym przez dwa bity mikrorozkazu: $S_{33} - S_{32}$. W tabeli 6 pokazano możliwe mikrooperacje wykonywane przez rejestr.

Tabela 6. Sterowanie rejestrem MN.

Sterowanie $S_{33} - S_{32}$	*Wejście* **clr**	*Wyjście* **MN_out**	*Opis*
X	0	00000000	Zerowanie asynchroniczne
00	1	**MN_out**	NOP
01	1	00000000	Zerowanie synchroniczne
10	1	**MN_in**	Ładowanie
11	1	**MN_out**	NOP

Multipleksery wyboru argumentu

W układzie znajdują się dwa multipleksery (**MUX A** oraz **MUX B**) dla wyboru argumentów ***A*** i ***B*** na wejściach układu **ALU**. Wejścia i wyjścia multiplekserów są 8-bitowe. Multiplekser argumentu ***B*** jest sterowany bitami $S_{27} - S_{26}$, a multiplekser argumentu ***A*** przez bity $S_{25} - S_{24}$. W tabeli 7 pokazano sposób wysterowania multiplekserów.

*Tabela 7: Sterowanie multiplekserami **MUXA** (bity $S_{25} - S_{24}$) oraz **MUXB** (bity $S_{27} - S_{26}$)*

Sterowanie	*Wyjście **MUX***	*Opis*
00	**MN_out**	Wybór argumentu **MN**
01	***MK_out***	Wybór argumentu **MK**
10	$S_{23} - S_{16}$	Argument natychmiastowy (odczytany z pamięci **ROM**)
11	**RW_out**	Wybór argumentu **RW**

Jednostka arytmetyczno-logiczna ALU

Jednostka arytmetyczno-logiczna ALU ma dwa 8-bitowe argumenty na wejściach ***A*** i ***B*** oraz 8-bitowe wyjście ***ALU_out***, na którym pojawia się wynik operacji. Trzybitowe wejście sterujące ($S_{31} - S_{29}$) służy do wyboru operacji wykonywanej przez **ALU**. W zależności od wyniku operacji wykonywanej przez ALU modyfikowane są bity ***z***, ***sign***, ***cout***, ***ovr***, i ***xor*** rejestru znaczników **RZ**. Wejście ***c_in*** wykorzystywane jest jedynie podczas operacji dodawania (***ALU_out*** = ***A*** + ***B*** + ***c_in***). W tabeli 8 zestawiono operacje wykonywane przez układ **ALU**.

Tabela 8: Sterowanie ALU.

Sterowanie **$S_{31} - S_{29}$**	*Operacja*	*Wyjście* **ALU_out**	*Opis*
000	NOP[1]	bez zmian	Wyjścia układu zachowują poprzednie wartości
001	ADD	***A*** + ***B*** + ***c_in***	Dodawanie (uaktualnia bity ***z***, ***sign***, ***cout***, ***ovr*** i ***xor*** rejestru znaczników **RZ**)
010	SUB	***A*** – ***B***	Odejmowanie (uaktualnia bity ***z***, ***sign***, ***cout***, ***ovr*** i ***xor*** rejestru znaczników **RZ**)
011	AND	***A*** AND ***B***	Operacja logiczna AND (uaktualnia bity ***z***, ***sign*** i ***xor*** rejestru znaczników **RZ**)
100	OR	***A*** OR ***B***	Operacja logiczna OR (uaktualnia bity ***z***, ***sign*** i ***xor*** rejestru znaczników **RZ**)
101	XOR	***A*** XOR ***B***	Operacja logiczna XOR (uaktualnia bity ***z***, ***sign*** i ***xor*** rejestru znaczników **RZ**)
110	NEG[2]	***~A***	Zmiana znaku argumentu ***A*** (uaktualnia bity ***z***, ***sign***, ***cout***, ***ovr*** i ***xor*** rejestru znaczników **RZ**)
111	INC	***A*** + 1	Inkrementacja argumentu A (uaktualnia bity ***z***, ***sign***, ***cout***, ***ovr*** i ***xor*** rejestru znaczników **RZ**)

Rejestr znaczników RZ

Rejestr znaczników **RZ** to rejestr 7-bitowy. Na kolejnych bitach tego rejestru zapisane są następujące wartości:

- ***z*** – ustawiony na wartość 1, gdy wynik operacji **ALU** jest równy zero i ustawiany na 0 w przeciwnym przypadku,
- ***sign*** – ustawiany na 1, gdy wartość wyniku **ALU** jest liczbą ujemną w kodzie U2,
- ***cout*** – zapisywana jest wartość wyjścia przeniesienia (pożyczki) bloku **ALU** podczas operacji dodawania/odejmowania,
- ***ovr*** – zapisywana jest wartość wyjścia nadmiaru bloku **ALU,** tj. sygnalizowane jest wystąpienie przekroczenia zakresu w operacji dodawania/odejmowania w kodzie U2,
- ***xor*** – zapisywana jest wartość operacji XOR najbardziej znaczących bitów argumentów wejściowych ***A*** oraz ***B***,
- ***P_{16}*** – zapisywany jest bit wysuwany z rejestru **RW** podczas przesuwania zawartości rejestru w lewo,
- ***P_0*** – zapisywana jest bit wysuwany z rejestru **MK** podczas przesuwania zawartości rejestru w prawo.

Na podstawie wartości bitów ***z*** i ***sign*** rejestru znaczników możliwe jest porównanie argumentów ***A*** i ***B***. Jeśli ***~z***&***sign*** = 1, to A<B. Jeśli ***~z***&***~sign*** = 1, to A>B. Tak obliczone wartości są podawane na wejścia 14 (***A<B***)i 15 (***A>B***) multipleksera **MUX W** w układzie sterowania.

[1] Blok ALU został uzupełniony o rejestr wewnętrzny podtrzymujący wynik operacji w ALU z poprzedniego taktu zegara. Ustawiając sterowanie 000 można, przez dowolny czas, utrzymywać na wyjściu ALU wynik interesującej nas operacji arytmetyczno-logicznej.

[2] Operacja NEG jest negacją liczby w kodzie U2. Ponieważ zakres liczb w kodzie U2 dla długości słowa 8 bitów wynosi od -128 do 127, to podczas negacji liczby -128 jest ustawiany bit ***ovr*** na wartość 1. Ponieważ operacja negacji jest równoznaczna operacji (0 – A), to dla A < 0 ustawiany jest bit ***cout*** na 1, a dla A>0 na 0.

2.3. Format mikrorozkazu

Mikrorozkazy to słowa 40 bitowe. Format mikrorozkazu przedstawia tabela 9.

Tabela 9. Przypisanie bitów mikrorozkazu poszczególnym mikrooperacjom.

Bity	*Znaczenie*
$S_5 - S_0$	Adres docelowy skoku
$S_9 - S_6$	Sterowanie multiplekserem skoku
$S_{12} - S_{10}$	Wartość ładowana do licznika pętli **LP**
$S_{15} - S_{13}$	Sterowanie licznika pętli **LP**
$S_{23} - S_{16}$	Argument natychmiastowy
$S_{25} - S_{24}$	Sterowanie multiplekserem argumentu ***A***
$S_{27} - S_{26}$	Sterowanie multiplekserem argumentu ***B***
S_{28}	Sterowanie multiplekserem wybierającym sygnał ***c_in***
$S_{29} - S_{31}$	Wybór operacji **ALU**
$S_{33} - S_{32}$	Sterowanie rejestrem **MN**
$S_{36} - S_{34}$	Sterowanie rejestrem **MK**
$S_{39} - S_{37}$	Sterowanie rejestrem **RW**

W tabeli 10 przedstawiono formularz ułatwiający kompletowanie mikrorozkazu w kodzie binarnym lub hexadecymalnym.

Tabela 10. Słowo mikrorozkazu.

RW			**MK**			**MN**		**ALU**			**S Alu**	**MUX B**		**MUX A**		**Argument natychmiastowy...**			
S_{39}	S_{38}	S_{37}	S_{36}	S_{35}	S_{34}	S_{33}	S_{32}	S_{31}	S_{30}	S_{29}	S_{28}	S_{27}	S_{26}	S_{25}	S_{24}	S_{23}	S_{22}	S_{21}	S_{20}

...Argument natychmiastowy				**LP ster**			**LP data**			**MUX W**				**Adres skoku**					
S_{19}	S_{18}	S_{17}	S_{16}	S_{15}	S_{14}	S_{13}	S_{12}	S_{11}	S_{10}	S_9	S_8	S_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0

W tabeli 11 przedstawiono przykładowy mikroprogram odejmowania dwóch liczb w naturalnym kodzie binarnym a w tabeli 12 ten mikroprogram w pliku tekstowym z rozszerzeniem *.mif*.

Tabela 11. Przykładowy mikroprogram odejmowania

RW			MK			MN		ALU			S Alu	MUX B		MUX A		Argument natychmiastowy...			
S_{39}	S_{38}	S_{37}	S_{36}	S_{35}	S_{34}	S_{33}	S_{32}	S_{31}	S_{30}	S_{29}	S_{28}	S_{27}	S_{26}	S_{25}	S_{24}	S_{23}	S_{22}	S_{21}	S_{20}
0	0	0	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0

...Argument natychmiastowy				LP ster			LP data			MUX W				Adres skoku					
S_{19}	S_{18}	S_{17}	S_{16}	S_{15}	S_{14}	S_{13}	S_{12}	S_{11}	S_{10}	S_9	S_8	S_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0

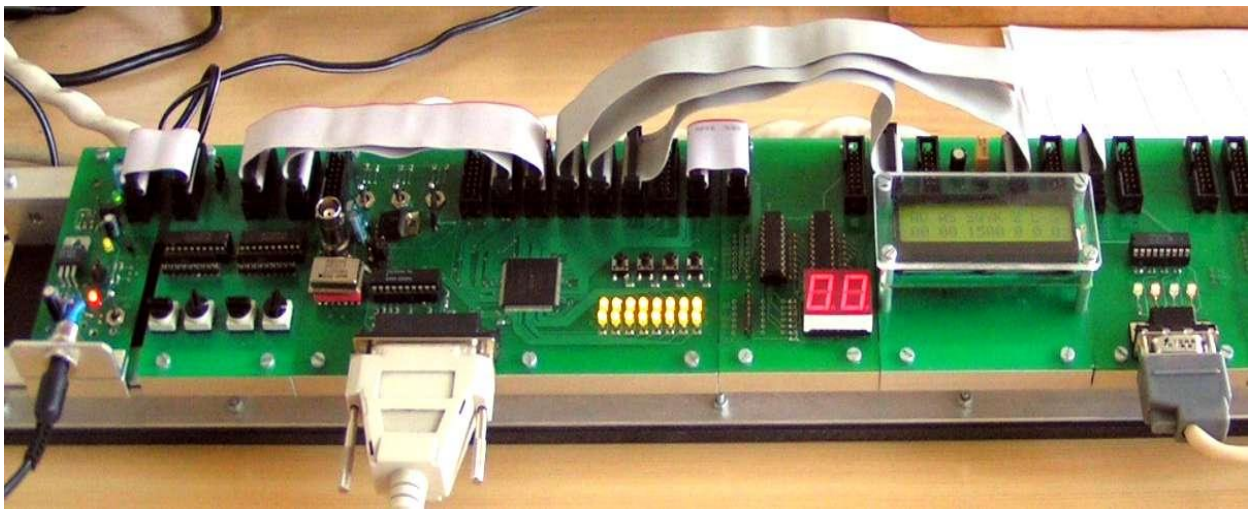
Tabela 12. Przykładowy mikroprogram odejmowania w pliku tekstowym sub.mif.

```
bin
0 : 00001010000100010000000000000000000000000
1 : 01000000010100010000000000000000000000000
2 : 00000000000100010000000000000000001000010
```

3. Zestaw laboratoryjny

Zestaw laboratoryjny przedstawiony na rysunku 4, składa się z siedmiu modułów SML3:

- moduł zasilacza **PS1**,
- moduł przełączników szesnastkowych **IN_4xHEX**,
- moduł FPDL EP1K30TC144, (**FPGA**)
- moduł wyświetlacza 7-segmentowego **7SEG2** (aby wyświetlić cały 16-bitowy wynik należy zastosować 2 moduły),
- moduł wyświetlacza **LCD** 182 (wersja z interfejsem 8-bitowym),
- moduł łącza szeregowego **EIA232_4**.

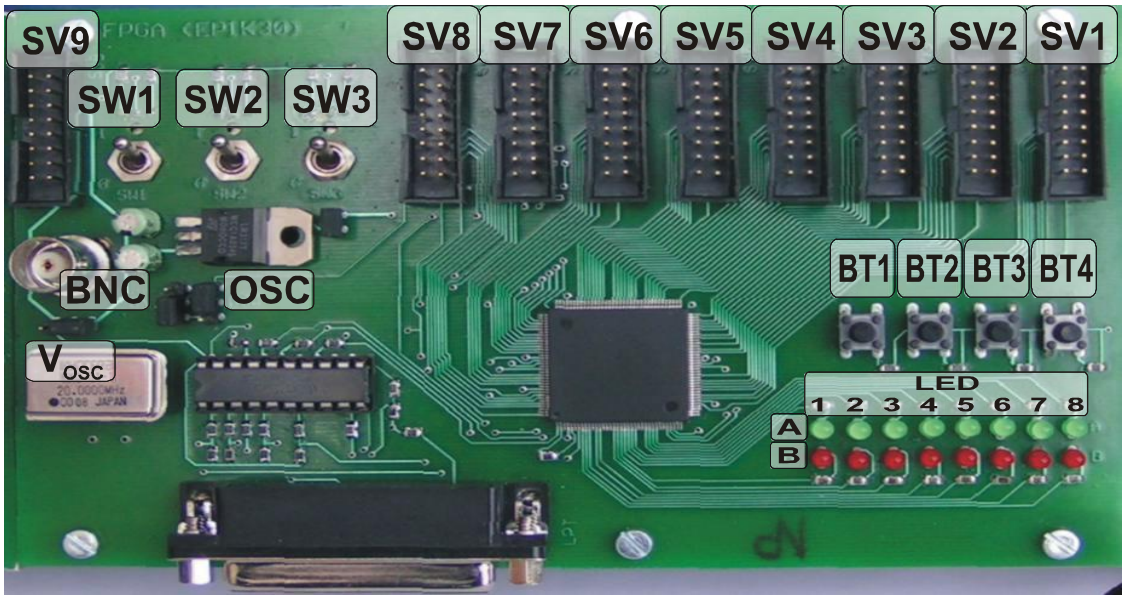


Rysunek 4.Widok zestawu laboratoryjnego.

Moduł FGPA

Moduł z układem FPGA zawiera:

- układ FPGA EP1k30,
- programator ByteBlasterMV ze złączem portu równoległego,
- generator kwarcowy 20MHz,
- gniazdo BNC,
- 9 gniazd w standardzie systemu SML3 (SV1- SV9),
- 16 diod LED,
- 4 przyciski (BT1-BT4),
- 3 przełączniki (SW1-SW3).



Rysunek 5. Pakiet z układem FPGA

Połączenia pomiędzy modułami systemu SML3:

Lp.	*Połączenie taśmą (moduł.złącze ↔ moduł.złącze)*	*Opis*
1	**PS1.*SV1*** ↔ **IN_4xHEX.*PWR***	Zasilanie układu
2	**IN_4xHEX.*OUTB*** ↔ **FPGA.*SV7***	Argument ***B***
3	**IN_4xHEX.*OUTA*** ↔ **FPGA.*SV6***	Argument ***A***
4	**FPGA.*SV1*** ↔ **7SEG2** **FPGA.*SV2*** ↔ **7SEG2**	Wyświetlenie wyniku (rejestr **RW**) Wyświetlenie wyniku (rejestr **MK**)
5	**FPGA.*SV3*** ↔ **LCD.*D8***	Sterowanie LCD
6	**FPGA.*SV4*** ↔ **LCD.*CTRL***	Sterowanie LCD
7	**FPGA.*SV5*** ↔ **EIA232_4.*SV1***	Ładowanie wartości pamięci RAM

Zwory (dostępne na pakiecie FPGA powinny mieć następujące ustawienie)

- OSC zwarte z CLK1 (ustawione w pozycji dolnej),
- BNC zwarte z CLK2 (ustawione w pozycji górnej),
- Vosc zwarte z +5V (ustawione w pozycji prawej),
- zwora 2.5V – zwarta (jeśli nie jest zwarta należy obowiązkowo zgłosić to prowadzącemu ćwiczenie).

Przełączniki (na pakiecie FPGA):

- SW1 – praca krokowa włączona/wyłączona,
- SW2 – krok pracy krokowej,
- SW3 – warunek zewnętrzny $wzew_1$, podawany na multiplekser warunku **MUX W**,
- BT1 – zerowanie układu.

Diody:

Na górnym rzędzie diod **LED A** o numerze **1** wyświetlany jest błąd transmisji przez łącze RS232, a na diodach od **2** do **8** wyświetlane są warunki rejestru znaczników **RZ**. Na diodzie drugiej znacznik ***z***, na kolejnych diodach są wyświetlane znaczniki ***sign***, ***cout***, ***ovr***, ***xor***, P_{16}, P_0.

Moduł wyświetlacza LCD

Dwuwierszowy wyświetlacz LCD 2x16 wraz ze sterownikiem ułatwia śledzenie pracy układu mikroprogramowanego. Dzięki niemu można zaprezentować stan układu w chwili bieżącej. W połączeniu z możliwością pracy krokowej umożliwia to dokładną obserwację działania układu. Jest to szczególnie cenne, gdy układ nie działa w zamierzony sposób.

Rysunek 6. Wyświetlacz LCD

Wyświetlacz LCD prezentuje stan układu z bieżącego cyklu zegarowego. Na wyświetlaczu prezentowane są następujące wielkości (począwszy od lewej strony):

- **AD** – adres, wartość wyjścia licznika adresowania **LA** (2 znaki),
- **AS** – adres skoku, bity S_4 – S_0 (2 znaki),
- **SWYK** – bity S_{39} – S_{24} słowa mikrorozkazu sterujące układem wykonawczym (4 znaki),
- **Z** – cztery pierwsze bity rejestru znaczników (***z***, ***sign***, ***cout***, ***ovr***),
- **L** – zawartość licznik pętli **LP** (1 znak),
- **S** – sterowanie multiplekserem warunku, bity S_9 – S_6 (1 znak).