MATEMATYKA I INFORMATYKA NA UCZELNIACH TECHNICZNYCH

MINUT 2019 (1), s. 24–35

Alina MOMOT[1], Robert TUTAJEWICZ[1]

[1]Wydział Automatyki, Elektroniki i Informatyki, Politechnika Śląska,
ul. Akademicka 16, 44-100 Gliwice

# Maszyna W – jak zaprojektować prosty rozkaz

**Streszczenie**. Budowa współczesnych komputerów jest niezwykle skomplikowana, jednakże w większości przypadków ich podstawowa architektura opiera się na założeniach sformułowanych przez Johna von Neumanna, Johna W. Mauchly'ego oraz Johna Presper Eckerta w 1945 roku. Tak więc, aby zrozumieć podstawowe zasady działania komputera wystarczy prosty model komputera zaprojektowany w ten sposób, aby spełniał te założenia.
W latach siedemdziesiątych ubiegłego wieku zespół pracowników Politechniki Śląskiej pod kierownictwem prof. Stefana Węgrzyna zaprojektował taki właśnie uproszczony model komputera wykładowego i nazwał go maszyną W. Niniejszy artykuł w skrótowy sposób opisuje budowę i sposób działania maszyny W oraz szczegółowo opisuje zasady projektowania prostych rozkazów dla tego modelowego komputera. W końcowej części artykułu zamieszczono również kilka przykładów zadań (projektów rozkazów) do samodzielnego wykonania wraz z rozwiązaniami.

## 1. Wstęp

Architektura komputera opisana przez Johna von Neumanna zakłada [2], że komputer jest urządzeniem składającym się z:

- **układu sterowania**, którego głównym zadaniem jest pobieranie instrukcji z pamięci oraz ich sekwencyjne przetwarzanie,
- **jednostki arytmetyczno-logicznej**, w której przetwarzane są dane,
- **pamięci**, w której przechowywane są dane oraz instrukcje programu,
- **urządzeń wejścia/wyjścia**, które służą do interakcji z otoczeniem.

Przy czym dwoma podstawowymi rejestrami wchodzącymi w skład układu sterowania są:

- **rejestr instrukcji**, który przechowuje kod wykonywanego rozkazu programu,
- **licznik rozkazów**, który przechowuje adres następnej instrukcji, która ma być wykonana w ramach programu.

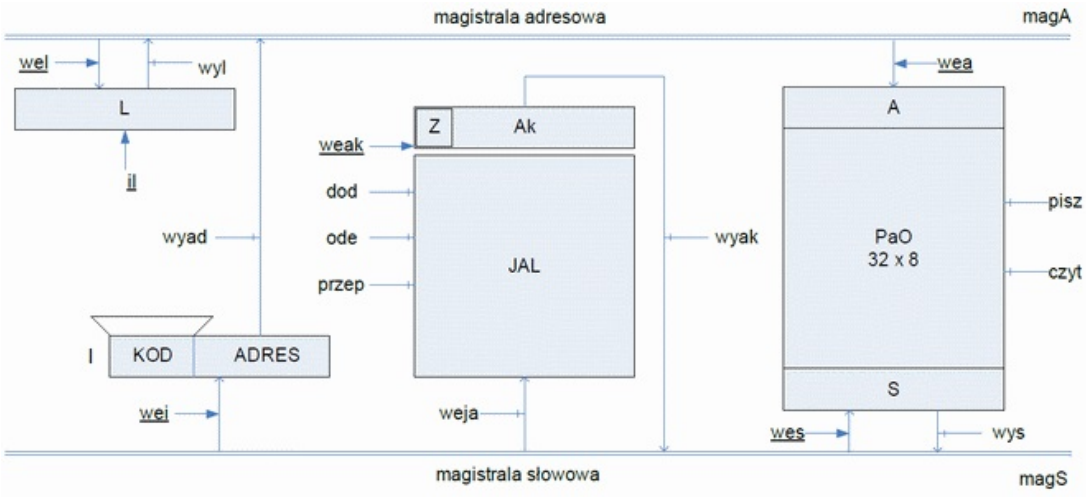
Autor korespondencyjny: A. Momot (alina.momot@polsl.pl).
Data wpłynięcia: 07.11.2019.

Aby przybliżyć zasady działania komputerów kolejnym pokoleniom tworzone są uproszczone modele komputerów. Przykładem tego może być między innymi Minivac 601 Digital Computer Kit – elektromechaniczny cyfrowy system komputerowy zaprojektowany w 1961 r. przez pioniera teorii informacji Claude'a Shannona jako zabawka edukacyjna ucząca zasad działania obwodów cyfrowych, czy też opisany przez Lindę Null i Julię Lobur prosty komputer MARIE (ang. *A Machine Architecture that is Really Intuitive and Easy*), za pomocą którego autorki prezentują przebieg cyklu rozkazu „pobierz, dekoduj, wykonaj" oraz pokazują sposób działania prawdziwych komputerów [4]. Autorki na stronie domowej książki udostępniają również symulator, który pozwala wykonywać programy napisane w asemblerze MARIE. W internecie dostępny jest również, na licencji BSD[1], program komputerowy SPIM – symulator procesora MIPS (ang. *Microprocessor without Interlocked Piped Stages*), napisany przez profesora Jamesa R. Larus'a i zaprojektowany do uruchamiania kodu języka asemblera dla tej architektury typu RISC (ang. *Reduced Instruction Set Computing*) [5]. Przy tej okazji warto też wspomnieć o witrynie internetowej *nand2tetris.org*, która to zawiera wszystkie materiały projektowe i narzędzia programowe niezbędne do zbudowania ogólnego systemu komputerowego od podstaw.

Również na Politechnice Śląskiej w zespole kierowanym przez prof. Węgrzyna powstał, w latach siedemdziesiątych ubiegłego wieku, projekt uproszczonego modelu komputera wykładowego mającego przybliżać studentom zasady działania komputerów [6]. Do dnia dzisiejszego, na wykładach z przedmiotu Podstawy Informatyki, studenci kierunku Informatyka zapoznają się na przykładzie tego modelu z podstawowymi zasadami działania komputerów [3]. Początkowo studenci szczegółowo poznawali te zasady ćwicząc podczas zajęć laboratoryjnych na modelu fizycznym – rzeczywistej implementacji projektu [1], aczkolwiek obecnie do ćwiczeń tych wykorzystywany jest programowy symulator maszyny W.

## 2. Podstawy budowy i działania maszyny W

Maszyna W to uproszczony model rzeczywistego komputera zaprojektowanego przy wykorzystaniu architektury von Nemanna. Schemat podstawowej wersji maszyny W przedstawia rysunek 1.




Rysunek 1. Schemat podstawowej wersji maszyny W

[1]Licencje BSD (Berkeley Software Distribution Licenses) zezwalają nie tylko na modyfikacje kodu źródłowego i jego rozprowadzanie w takiej postaci, ale także na rozprowadzanie produktu bez postaci źródłowej czy włączenia do zamkniętego oprogramowania, pod warunkiem załączenia do produktu informacji o autorach oryginalnego kodu i treści licencji.

W schemacie maszyny W można wyróżnić kilka podstawowych składowych, takich jak:

- pamięć operacyjna (PaO), wraz z wyróżnionymi rejestrami: adresu A oraz słowa S,
- jednostka arytmetyczno-logiczna (JAL), wraz z wyróżnionym rejestrem Akumulatora Ak i wchodzącym w jego skład bitem znaku Z,
- układ sterujący, w którego skład wchodzą rejestr instrukcji I oraz licznika rozkazów L.

Elementy te połączone są poprzez magistralę słowową (zwaną również magistralą danych) i magistralę adresową. Przesyłanie danych miedzy tymi elementami jest sterowane za pomocą kilkunastu sygnałów binarnych zwanych sygnałami sterującymi. Aby zrozumieć działanie całego układu należy najpierw zapoznać się z działaniem każdego z elementów składowych oraz rolą, jaką pełnią poszczególne sygnały sterujące.

## 2.1. Pamięć operacyjna

Pamięć operacyjna (PaO) przechowuje program w postaci ciągu rozkazów komputera oraz dane, na których ten program jest wykonywany. W dowolnym momencie obliczeń, na życzenie procesora, pamięć udostępnia przechowywane przez siebie dane i rozkazy. Procesor ma także możliwość zapisywania w pamięci wyników wykonywanych przez siebie obliczeń.

Pamięć składa się z komórek, z których każda przechowuje pojedyncze słowo i jest identyfikowana poprzez numer komórki nazywany adresem. Aby wykonać jakąkolwiek operację na pamięci, procesor musi zawsze podać adres komórki, której ta operacja dotyczy.

Aby odczytać określoną komórkę pamięci, należy najpierw wpisać adres komórki do rejestru adresowego pamięci A przy użyciu sygnału *wea*. Aktywacja tego sygnału umożliwia wpisanie do rejestru adresowego pamięci A adresu przesyłanego magistralą adresową. W następnym kroku możliwe jest odczytanie odpowiedniej komórki z pamięci (operację aktywuje sygnał *czyt*, który wprowadza do rejestru słowa S odczytaną wartość) i przesłanie odczytanej wartości na magistralę danych za pomocą sygnału *wys*.

Aby zapisać coś w pamięci, należy najpierw wpisać adres odpowiedniej komórki pamięci w rejestrze adresowym A (sygnał *wea*) oraz wprowadzić do rejestru słowowego pamięci S wartość, jaka ma być zapisana w pamięci przy użyciu sygnału *wes*. Aktywacja tego sygnału umożliwia przesłanie zawartości magistrali danych do rejestru S. W następnym kroku należy aktywować sygnał *pisz*, który spowoduje zapisanie zawartości rejestru S w komórce pamięci, której adres znajduje się w rejestrze A.

## 2.2. Jednostka arytmetyczno-logiczna

Jednostka arytmetyczno-logiczna (JAL) jest elementem procesora odpowiedzialnym za wykonywanie obliczeń takich jak dodawanie, odejmowanie, suma i iloczyn logiczny lub przesunięcia bitowe. Zakłada się przy tym, że wynik obliczeń zostanie zapamiętany w rejestrze akumulatora Ak, będącym częścią składową JAL.

Ponieważ jednostka arytmetyczno-logiczna w maszynie W ma tylko jedno wejście, podczas wykonywania operacji dwuargumentowych pierwszy argument, potrzebny do wykonania operacji, zostaje pobrany z akumulatora. Zatem należy pamiętać, aby przed wykonaniem właściwej operacji dwuargumentowej zadbać o odpowiednie wypełnienie akumulatora (poprzez uprzednie wprowadzenie do akumulatora wartości pierwszego argumentu planowanej do wykonania operacji). Można to zrealizować przy użyciu sygnału

*przep*, który aktywuje operację przepisywania stanu wejścia JAL. Do poprawnego wykonania operacji wprowadzenia wartości argumentu do rejestru akumulatora niezbędna jest również aktywacja sygnału *weja*, który umożliwia wprowadzenie na wejście JAL wartości podanej na magistralę danych, oraz sygnału *weak*, który umożliwia wprowadzenie wypracowanej w jednostce arytmetyczno-logicznej wartości do rejestru akumulatora. Podsumowując, aby wypełnić rejestr akumulatora wartością podaną na magistralę danych, konieczna jest aktywacja trzech sygnałów: *weja*, *przep* oraz *weak*.

W najprostszej wersji maszyny W (widocznej na rysunku 1), jednostka arytmetyczno-logiczna potrafi wykonywać tylko trzy operacje: przepisywanie stanu podanego na wejście (aktywowane sygnałem *przep*), dodawanie arytmetyczne[2] (aktywowane sygnałem *dod*) i odejmowanie[3] (aktywowane sygnałem *ode*). Sygnał *weja* powoduje udostępnienie wartości będącej na magistrali danych na wejście JAL, zaś sygnał *weak* pozwala wynik obliczeń zapisać do rejestru akumulatora. Zawartość rejestru akumulatora można wyprowadzić na magistralę danych przy użyciu sygnału *wyak*.

Obliczenia w JAL wykonywane są na liczbach binarnych zapisanych w kodzie uzupełnieniowym do $2^4$. W dowolnym momencie obliczeń możliwe jest sprawdzenie, czy w akumulatorze znajduje się liczba ujemna, czy też nieujemna. W tym celu wystarczy sprawdzić bit znaku liczby przechowywanej w akumulatorze (nazywany sygnałem stanu Z). Jeśli liczba w akumulatorze jest liczbą ujemną, bit Z będzie miał wartość jeden. W przeciwnym przypadku bit ten będzie równy zero. Układ sterujący procesora może sterować wykonywaniem obliczeń w różny sposób w zależności od stanu bitu Z.

## 2.3. Układ sterujący

Układ sterujący, na podstawie stanu maszyny W (czyli zawartości poszczególnych rejestrów, tworzących ten układ), określa sygnały sterujące, jakie w danym momencie mają być aktywne. Rejestrami wpływającymi na działanie układu sterującego oprócz akumulatora są rejestr instrukcji I oraz rejestr licznika rozkazów L.

Licznik rozkazów przechowuje adres kolejnego rozkazu do wykonania. Ponieważ najczęściej kolejne, następujące po sobie rozkazy znajdują się w kolejnych komórkach pamięci, do licznika rozkazów dodano sygnał *il*, którego zadaniem jest inkrementacja (czyli zwiększenie o 1) zawartości tego rejestru. Ponadto z licznikiem rozkazów związane są sygnały: *wel*, który aktywuje operację zapisania w rejestrze L aktualnej zawartości magistrali adresowej oraz *wyl*, który powoduje wyprowadzenie zawartości rejestru na tę magistralę.

W rejestrze instrukcji przez większość czasu wykonywania rozkazu znajduje się słowo opisujące wykonywany rozkaz. Rejestr ten można podzielić na dwie części: kodową i adresową (patrz rysunek 1). Pierwsza część rejestru przechowuje kod rozkazu, zaś druga argument. Przy czym najczęściej argumentem jest adres komórki, do której odwołuje się dany rozkaz, stąd też nazwa drugiej części (część adresowa) i oznaczenie symboliczne Ad.

Z rejestrem instrukcji związane są dwa sygnały: *wyad* oraz *wei*. Pierwszy z nich, sygnał *wyad* pozwala wyprowadzić argument (zawartość części adresowej rejestru I) na magistralę adresową. Drugi, sygnał *wei* pozwala na przepisanie słowa opisującego rozkaz do wykonania z magistrali danych do rejestru I. Warto w tym miejscu podkreślić fakt, że szerokość (pojemność) magistrali adresowej i magistrali danych są

[2]Do zawartości akumulatora dodawana jest wartość podana na wejście JAL.

[3]Od zawartości akumulatora odejmowana jest wartość podana na wejście JAL.

[4]Jest to system reprezentacji liczb całkowitych w dwójkowym systemie pozycyjnym. Jego nazwa wzięła się ze sposobu obliczania liczb przeciwnych: wartości przeciwne liczb n-bitowych uzyskujemy poprzez odjęcie tej liczby od $2^n$ (uzupełnienie do liczby $2^n$). Zatem na $n$ bitach zapisujemy liczby z zakresu od $-2^{n-1}$ do $2^{n-1}-1$, przy czym pierwszy, najstarszy bit, traktujemy jako tak zwany bit znaku. Jego wartość równa 1 oznacza liczbę ujemną, zaś 0 dodatnią.

różne. W przypadku najprostszej wersji maszyny W, magistralą danych można przesyłać słowa 8-bitowe, natomiast magistralą danych – słowa 5-bitowe.

## 2.4. Sygnały sterujące: impulsowe i poziomowe

Podczas analizy pracy maszyny W należy zwrócić uwagę na fakt, że układ sterujący jest układem sekwencyjnym synchronicznym, zatem wymaga do prawidłowej pracy sygnału zegarowego (taktującego). Wszystkie operacje wewnątrz jednostki centralnej jak i operacje w całym systemie odbywają się w takt sygnału zegarowego, który to sygnał powinien być doprowadzony do wszystkich elementów systemu. Częstotliwość sygnału zegarowego ma bezpośredni wpływ na szybkość operacji wykonywanych przez procesor, układy pamięci czy też urządzenia wejścia/wyjścia. Odwrotność częstotliwości generatora zegarowego to okres sygnału zegarowego nazywany zwykle cyklem zegarowym lub taktem zegara.

Na schemacie maszyny W widocznym na rysunku 1 można wyróżnić 16 sygnałów sterujących. Sygnały te można podzielić na dwie kategorie:

- sygnały impulsowe – bardzo krótkie sygnały aktywowane na koniec taktu sygnału zegarowego (wyróżnione na rysunku poprzez podkreślenie ich nazw) oraz
- sygnały poziomowe – długie sygnały, aktywne przez całą długość taktu sygnału zegarowego.

Sygnały poziomowe służą przede wszystkim do aktywowania operacji wyprowadzania danych zapisanych w rejestrach na magistrale, natomiast sygnały impulsowe używane są w celu wpisywania nowych danych do rejestrów. Wynika to z faktu iż wyprowadzając daną wartość z rejestru na magistralę musi minąć pewien czas, aby pojawiająca się na magistrali dana wartość mogła być traktowana jako wiarygodna (stabilny poziom napięcia). Dopiero wtedy (pod koniec taktu zegarowego) można tę daną wartość przepisać do kolejnego rejestru.

Operacje aktywowane poszczególnymi sygnałami sterującymi można opisać używając symboliki przesyłów międzyrejestrowych[5]:

- dla sygnałów poziomowych
  1. $wyl : (L) \rightarrow magA$ – wyprowadzenie zawartości rejestru licznika na magistralę adresową,
  2. $wyad : (Ad) \rightarrow magA$ – wyprowadzenie zawartości części adresowej rejestru instrukcji na magistralę adresową,
  3. $wyak : (Ak) \rightarrow magS$ – wyprowadzenie zawartości rejestru akumulatora na magistralę danych,
  4. $wys : (S) \rightarrow magS$ – wyprowadzenia zawartości rejestru słowa pamięci operacyjnej na magistralę danych,
  5. $pisz : (S) \rightarrow (A)$ – wpisanie zawartości rejestru słowa S pamięci operacyjnej do komórki pamięci o adresie zawartym w rejestrze adresowym A pamięci operacyjnej,
  6. $czyt : ((A)) \rightarrow S$ – wpisanie zawartości komórki pamięci o adresie zawartym w rejestrze adresowym A pamięci operacyjnej do rejestru słowa S,
  7. $weja : (magS) \rightarrow JAL$ – wpisanie wartości podanej na magistralę danych na wejście jednostki artymetyczno-logicznej,

[5] Zakładając, że R oznacza rejestr o nazwie R, zapis (R) oznacza zawartość rejestru R, natomiast ((R)) zawartość komórki pamięci o adresie wpisanym do rejestru R.

8. *przep* – przepisanie danej podanej na na wejście jednostki artymetyczno-logicznej,
9. *dod* – dodanie do zawartości rejestru akumulatora wartości podanej na wejście JAL,
10. *ode* – odjęcie od zawartości rejestru akumulatora wartości podanej na wejście JAL,

- dla sygnałów impulsowych

1. $wel : (magA) \rightarrow L$ – wprowadzenie zawartości magistrali adresowej do rejestru licznika,
2. $wea : (magA) \rightarrow A$ – wprowadzenie zawartości magistrali adresowej do rejestru adresowego pamięci operacyjnej,
3. $wes : (magS) \rightarrow S$ – wprowadzenie zawartości magistrali danych do rejestru słowa pamięci operacyjnej,
4. $wei : (magS) \rightarrow I$ – wprowadzenie zawartości magistrali danych do rejestru instrukcji,
5. $weak : (JAL) \rightarrow Ak$ – wprowadzenie wypracowanej w jednostce arytmetyczno-logicznej wartości do rejestru akumulatora,
6. $il : (L) + 1 \rightarrow L$ – inkrementacja zawartości rejestru licznika.

# 3. Projektowanie rozkazów

Jednym z etapów projektowania procesora jest ustalenie listy rozkazów realizowanych przez dany procesor oraz zdefiniowanie, dla każdego rozkazu, sekwencji działań potrzebnych do jego wykonania. Pod pojęciem projektowania rozkazu będziemy rozumieć ustalenie przebiegów sygnałów sterujących, aktywujących wykonanie wszystkich przesyłów międzyrejestrowych, składających się na dany rozkaz.

Każdy rozkaz wykonywany jest etapami. Pierwszy etap to pobranie i zdekodowanie rozkazu. Dopiero po zakończeniu tego etapu procesor jest w stanie określić jaki rozkaz wykonuje. Oznacza to, że etap pobrania i dekodowania musi być wykonywany identycznie dla wszystkich projektowanych rozkazów. W maszynie W wykonanie tego etapu realizowane jest jako pierwsza faza rozkazu i składają się na nią sygnały: *czyt* (odczytanie komórki pamięci), *wys* (przesłanie zawartości odczytanej komórki na magistralę danych), *wei* (wprowadzenie odczytanej zawartości do rejestru instrukcji) i *il* (inkrementacja licznika rozkazów). Przy czym zakłada się, że w momencie odczytu zawartości komórki pamięci operacyjnej, w rejestrze adresowym pamięci operacyjnej A oraz rejestrze licznika znajduje się ta sama wartość – adres mającego się wykonać rozkazu. Należy też podkreślić, że w poprawnie zaprojektowanym rozkazie sygnał *wei* występuje zawsze tylko i wyłącznie w pierwszej fazie rozkazu (zatem przez wszystkie kolejne fazy potrzebne do zrealizowania rozkazu jego kod znajduje się w części kodowej rejestru instrukcji).

W kolejnych fazach realizowane są dalsze etapy wykonania rozkazu. Są to: ustalenie adresu efektywnego argumentu, jego odczyt i wykonanie na nim właściwej operacji oraz przygotowanie do wykonania następnego rozkazu (zadbanie o to, aby w rejestrze adresowym pamięci operacyjnej A oraz rejestrze licznika znajdowała się ta sama wartość – adres kolejnego rozkazu do wykonania). Cały proces projektowania rozkazów zostanie przedstawiony na przykładzie opisanych poniżej rozkazów DOD, ŁAD oraz SOM.

## 3.1. Rozkaz DOD

Działanie rozkazu DOD można opisać symbolicznie za pomocą przesyłu międzyrejestrowego:

$$(Ak) + ((Ad)) \rightarrow Ak.$$

W wyniku działania tego rozkazu do rejestru akumulatora jednostki arytmetyczno-logicznej należy dodać zawartość komórki pamięci, której adres jest podany jako argument projektowanego rozkazu i znajduje się w części adresowej rejestru instrukcji (po pobraniu i zdekodowaniu rozkazu). Wynik dodawania należy zapisać w rejestrze akumulatora.

Podobnie jak w przypadku każdego innego rozkazu, projektowanie rozkazu zaczniemy od fazy pobrania i dekodowania rozkazu. Najpierw należy odczytać rozkaz z pamięci i przesłać go do rejestru instrukcji oraz zwiększyć o 1 zawartość licznika rozkazów. Wykonanie tych działań wymaga uaktywnienia sygnałów $czyt, wys, wei$ oraz $il$. Te właśnie sygnały będą tworzyć pierwszą fazę wykonania rozkazu dodawania.

Po zdekodowaniu konkretnego rozkazu można przystąpić do realizacji charakterystycznych dla niego działań. W tym przypadku do akumulatora musimy dodać zawartość komórki pamięci, której adres jest argumentem rozkazu. Samo dodawanie musi być poprzedzone odczytaniem zawartości odpowiedniej komórki pamięci. Należy przy tym pamiętać, że przy aktywacji sygnału $czyt$ zawsze odczytywana jest komórka, której adres znajduje się w rejestrze adresowym układu pamięci (rejestrze A). Tymczasem adres komórki, którą mamy odczytać znajduje się w części adresowej (Ad) rejestru instrukcji. Przed odczytaniem z pamięci należy zatem przesłać zawartość rejestru Ad do rejestru A, co można zapisać symbolicznie jako $(Ad) \rightarrow A$. Przesył taki będzie wykonany po uaktywnieniu sygnałów $wyad$ (wyprowadzenie zawartości części adresowej rejestru instrukcji na magistralę adresową) i $wea$ (zapisanie zawartości magistrali adresowej w rejestrze A układu pamięci). Te dwa sygnały ($wyad$ i $wea$) tworzą drugą fazę realizacji rozkazu dodawania.

Wszystkie pozostałe czynności zostaną wykonane w trzeciej fazie wykonania tego rozkazu. A będą to:
– odczytanie argumentu z pamięci (aktywowane sygnałem $czyt$),
– przesłanie go do jednostki arytmetyczno-logicznej (co wymaga sygnałów $wys$ i $weja$),
– wykonanie dodawania (sygnał $dod$) i zapisanie jego wyniku w akumulatorze (sygnał $weak$).
Czynności te można zapisać symbolicznie jako $(Ak) + ((A)) \rightarrow Ak$. Dodatkowo, ponieważ trzecia faza będzie ostatnią fazą realizacji rozkazu dodawania, musimy pamiętać o przesłaniu adresu następnego rozkazu do wykonania z rejestru licznika rozkazów L do rejestru adresowego pamięci A, co można zapisać symbolicznie jako $(L) \rightarrow A$. Ten przesył zostanie zrealizowany dzięki sygnałom $wyl$ i $wea$.

Warto przy tym też zwrócić uwagę na pewną optymalizację działań. Z uwagi na fakt iż przesył $(Ak) + ((A)) \rightarrow Ak$ jest niezależny od przesyłu $(L) \rightarrow A$, a transfer danych w każdym z nich zajmuje inną magistralę (odpowiednio magistralę danych i adresową), operacje związane z tymi przesyłami mogą być wykonane w tym samym takcie zegarowym.

Widzimy więc, że rozkaz dodawania DOD tworzą trzy fazy (takty zegara), w których muszą być aktywne następujące sygnały sterujące:

1. *czyt wys wei il*;
2. *wyad wea*;
3. *czyt wys weja dod weak wyl wea.*

## 3.2. Rozkaz ŁAD

Działanie rozkazu ŁAD można opisać symbolicznie za pomocą przesyłu międzyrejestrowego:

$$(Ak) \rightarrow (Ad).$$

W wyniku działania tego rozkazu zawartość rejestru akumulatora jednostki arytmetyczno-logicznej należy wpisać do komórki pamięci, której adres jest podany jako argument projektowanego rozkazu i znajduje się w części adresowej rejestru instrukcji (po pobraniu i zdekodowaniu rozkazu).

Jak wspomniano uprzednio pierwszy takt każdego rozkazu rozpoczyna się od fazy pobrania i dekodowania rozkazu. Należy zatem odczytać rozkaz z pamięci i przesłać go do rejestru instrukcji oraz zwiększyć o 1 zawartość licznika rozkazów. Wykonanie tych działań wymaga uaktywnienia sygnałów $czyt, wys, wei$ oraz $il$. Te właśnie sygnały będą tworzyć pierwszą fazę wykonania rozkazu.

Po zdekodowaniu rozkazu można przystąpić do charakterystycznych dla niego działań. Ponieważ aktywny sygnał *pisz* powoduje zapis wartości zapisanej w rejestrze słowa S pamięci operacyjnej do komórki pamięci o adresie zawartym w rejestrze adresowym A pamięci operacyjnej, przed wykonaniem operacji zapisu należy odpowiednio wypełnić rejestry uczestniczące w tej operacji. Zatem do rejestru S należy przesłać zawartość rejestru akumulatora, co można symbolicznie zapisać jako $(Ak) \to S$. Przesył taki będzie wykonany po uaktywnieniu sygnałów *wyak* (wyprowadzenie zawartości rejestru akumulatora na magistralę danych) oraz *wes* (zapisanie zawartości magistrali danych w rejestrze S układu pamięci). Należy także przesłać zawartość części adresowej Ad rejestru instrukcji do rejestru adresowego pamięci operacyjnej A, co można zapisać symbolicznie jako $(Ad) \to A$. Przesył taki będzie wykonany po uaktywnieniu sygnałów *wyad* (wyprowadzenie zawartości części adresowej rejestru instrukcji na magistralę adresową) i *wea* (zapisanie zawartości magistrali adresowej w rejestrze adresowym układu pamięci).

Z uwagi na fakt iż oba przesyły $(Ak) \to S$ oraz $(Ad) \to A$ są niezależne od siebie, a transfer danych w każdym z nich zajmuje inną magistralę (odpowiednio magistralę danych i adresową), operacje związane z tymi przesyłami mogą być wykonane w tym samym takcie zegarowym. Odbywać się to będzie drugim takcie naszego rozkazu, który będą tworzyć sygnały $wyak, wes$ oraz $wyad, wea$.

Po wpisaniu odpowiednich wartości do rejestru słowa i rejestru adresowego pamięci operacyjnej, w kolejnym (trzecim) takcie należy aktywować sygnał zapisu *pisz*. Dodatkowo, ponieważ trzecia faza będzie ostatnią fazą realizacji rozkazu, musimy pamiętać o przesłaniu adresu następnego rozkazu do wykonania z rejestru licznika rozkazów L do rejestru adresowego pamięci A, co można zapisać symbolicznie jako $(L) \to A$. Ten przesył zostanie zrealizowany dzięki sygnałom *wyl* i *wea*.

Widzimy więc, że rozkaz ŁAD tworzą trzy fazy (takty zegara), w których muszą być aktywne następujące sygnały sterujące:

1. *czyt wys wei il*;
2. *wyak wes wyad wea*;
3. *pisz wyl wea*.

## 3.3. Rozkaz SOM

Działanie rozkazu SOM można opisać symbolicznie za pomocą przesyłu międzyrejestrowego:

$$\text{jeżeli Z=1, to} \quad (Ad) \to L.$$

W wyniku działania tego rozkazu zawartość części adresowej rejestru instrukcji powinna zostać wpisana do rejestru licznika, o ile tylko zawartość rejestru akumulatora jednostki arytmetyczno-logicznej jest ujemna. Oznacza to, że rozkaz ten powoduje ustalenie adresu następnego do realizacji rozkazu w zależności od zawartości akumulatora. Jeżeli w akumulatorze jest liczba nieujemna, następnym wykonywanym rozkazem ma być rozkaz znajdujący się w kolejnej komórce pamięci. Natomiast jeżeli zawartość akumulatora jest ujemna, następnym rozkazem do wykonania ma być ten, który zapisano w komórce pamięci o adresie podanym jako argument rozkazu.

Jak zwykle pierwszą fazę projektowanego rozkazu tworzą sygnały $czyt, wys, wei, il$. Następnie należy sprawdzić, czy w akumulatorze znajduje się wartość ujemna. Jest to możliwe dzięki istnieniu odpowiednich sygnałów stanu procesora. W tym wypadku interesuje nas wartość bitu znaku Z rejestru akumulatora,

czyli znak liczby przechowywanej w akumulatorze. Sygnał stanu Z informuje procesor czy w akumulatorze jest liczba ujemna (wtedy bit ten przyjmuje wartość 1).

Rozkaz SOM jest więc rozkazem skoku warunkowego. Jeżeli Z = 1, to w akumulatorze jest liczba ujemna, a to oznacza, że należy wykonać skok, czyli przesłać do rejestrów L i A adres znajdujący się w części adresowej rejestru instrukcji, co można zapisać symbolicznie jako $(Ad) \rightarrow L, A$. Wymaga to uaktywnienia sygnałów $wyad, wel$ oraz $wea$. Warto przy tym podkreślić, że przesył $(Ad) \rightarrow A$ realizowany równocześnie z $(Ad) \rightarrow L$ wynika z optymalizacji, czyli dążenia do uzyskania pożądanego efektu w minimalnej liczbie taktów zegarowych. Zamiast więc realizować dwa przesyły: najpierw $(Ad) \rightarrow L$, a potem $(L) \rightarrow A$, wykorzystujemy fakt, że w obu wypadkach przesyłana jest ta sama wartość, po tej samej magistrali.

Jeżeli Z = 0, to w akumulatorze jest liczba nieujemna, więc jedynie musimy zakończyć nasz bieżący rozkaz, przygotowując w rejestrze adresowym pamięci $A$ adres kolejnego rozkazu. Adres następnego rozkazu znajduje się w rejestrze licznika L (po wykonaniu inkrementacji w pierwszej fazie), zatem wystarczy go przesłać do rejestru $A$. Z tym przesyłem związane są sygnały $wyl, wea$.

Widzimy więc, że rozkaz dodawania SOM tworzą dwie fazy (takty zegara), w których muszą być aktywne następujące sygnały sterujące:

1. *czyt wys wei il*;
   Jeżeli Z=1
2. *wyad wel wea*;
   w przeciwnym przypadku
2.' *wyl wea.*

# 4. Zadania do samodzielnego wykonania

Na zakończenie proponuje się Czytelnikowi samodzielne wykonanie kilku zadań, sformułowanych w dalszej części niniejszej sekcji, w celu utrwalenia opisywanych wyżej wiadomości i zasad projektowych. Pod treścią zadań zamieszczono ich rozwiązania w postaci opisu działania rozkazu w języku naturalnym oraz wyszczególniono nazwy niezbędnych sygnałów sterujących, które należy aktywować w poszczególnych fazach rozkazu. Przy czym sugeruje się czytelnikowi próbę samodzielnego zmierzenia się ze zrozumieniem symboliki rozkazu, wskazania aktywnych sygnałów sterujących oraz zastanowienia się nad optymalizacją czasową proponowanego rozwiązania, a dopiero potem, na koniec, porównanie swoich przemyśleń z gotowym (optymalnym) rozwiązaniem.

## 4.1. Treści zadań

Zaprojektować rozkazy opisane w sposób symboliczny jako:

1. $2 * (Ak) \rightarrow Ak$,
2. $3 * (Ak) \rightarrow Ak$,
3. $5 * (Ak) \rightarrow Ak$,
4. $8 * (Ak) \rightarrow Ak$,
5. $-(Ak) \rightarrow Ak$,

6. $((L)+1) \rightarrow (Ad)$,
7. jeżeli $Z = 1$, to $(Ak) - ((Ad)) \rightarrow Ak$,
8. jeżeli $Z = 1$, to $2 * (Ak) \rightarrow Ak, (Ad)$,
9. jeżeli $((Ad)) < 0$, to $0 \rightarrow Ak, (Ad)$;
   w przeciwnym przypadku $((Ad)) \rightarrow Ak$,
10. jeżeli $(Ak) < ((Ad))$, to $(Ak) - 2 * ((Ad)) \rightarrow Ak$.

## 4.2. Rozwiązania zadań

1. Rozkaz powinien zrealizować podwojenie zawartości akumulatora. Do jego wykonania potrzebne są 2 fazy (takty zegara), w których muszą być aktywne następujące sygnały sterujące:
   1. *czyt wys wei il*;
   2. *wyak weja dod weak wyl wea.*
2. Rozkaz powinien zrealizować potrojenie zawartości akumulatora. Do jego wykonania potrzebne są 3 fazy (takty zegara), w których muszą być aktywne następujące sygnały sterujące:
   1. *czyt wys wei il*;
   2. *wyak weja dod weak wes*;
   3. *wys weja dod weak wyl wea.*
3. Rozkaz powinien zrealizować pięciokrotone zwiększenie zawartości akumulatora. Do jego wykonania potrzebne są 4 fazy (takty zegara), w których muszą być aktywne następujące sygnały sterujące:
   1. *czyt wys wei il*;
   2. *wyak weja dod weak wes*;
   3. *wyak weja dod weak*;
   4. *wys weja dod weak wyl wea.*
4. Rozkaz powinien zrealizować ośmiokrotne zwiększenie zawartości akumulatora. Do jego wykonania potrzebne są 4 fazy (takty zegara), w których muszą być aktywne następujące sygnały sterujące:
   1. *czyt wys wei il*;
   2. *wyak weja dod weak*;
   3. *wyak weja dod weak*;
   4. *wyak weja dod weak wyl wea.*
5. W wyniku działania rozkazu wartość zawarta akumulatorze powinna zostać zmieniona na przeciwną. Do jego wykonania potrzebne są 3 fazy (takty zegara), w których muszą być aktywne następujące sygnały sterujące:
   1. *czyt wys wei il*;
   2. *wyak weja ode weak wes*;
   3. *wys weja ode weak wyl wea.*

6. W wyniku działania rozkazu w pamięci, pod adresem podanym argumentem rozkazu, powinna zostać wpisana wartość zapisana w kolejnej komórce pamięci operacyjnej względem komórki pamięci, w której znajduje się dany rozkaz. Do jego wykonania potrzebne są 4 fazy (takty zegara), w których muszą być aktywne następujące sygnały sterujące:
    1. *czyt wys wei il*;
    2. *wyl wea*;
    3. *czyt wyad wea il*[6];
    4. *pisz wyl wea.*

7. Rozkaz powinien zrealizować odejmowanie od zawartości akumulatora zawartości komórki pamięci, której adres podany jest argumentem rozkazu, o ile tylko zawartość rejestru akumulatora jednostki arytmetyczno-logicznej jest ujemna. Do jego wykonania potrzebne są maksymalnie 3 fazy (takty zegara), w których muszą być aktywne następujące sygnały sterujące:
    1. *czyt wys wei il*;
        Jeżeli Z=1
    2. *wyad wea*;
    3. *czyt wys weja ode weak wyl wea*;
        w przeciwnym przypadku
    2.' *wyl wea.*

8. Rozkaz powinien zrealizować podwojenie zawartości akumulatora, a następnie wyznaczoną wartość pozostawić w akumulatorze i wpisać do komórki pamięci, której adres podany jest argumentem rozkazu, o ile tylko zawartość rejestru akumulatora jednostki arytmetyczno-logicznej jest ujemna. Do jego wykonania potrzebne są maksymalnie 4 fazy (takty zegara), w których muszą być aktywne następujące sygnały sterujące:
    1. *czyt wys wei il*;
        Jeżeli Z=1
    2. *wyak weja dod weak*;
    3. *wyak wes wyad wea*;
    4. *pisz wyl wea*;
        w przeciwnym przypadku
    2.' *wyl wea.*

9. W wyniku działania rozkazu wartość zero należy wpisać do akumulatora i do komórki pamięci, której adres podany jest argumentem rozkazu, o ile tylko zawartość tej komórki pamięci była ujemna. W przeciwnym przypadku do akumulatora należy wpisać zawartość komórki pamięci, której adres podany jest argumentem rozkazu. Do jego wykonania potrzebne jest maksymalnie 6 faz (taktów zegara), w których muszą być aktywne odpowiednio sygnały sterujące:
    1. *czyt wys wei il*;
    2. *wyad wea*;
    3. *czyt wys weja przep weak wyl wea*;

[6]Aktywny sygnał *il* w tej fazie rozkazu świadczy o tym, że projektując ten rozkaz zakładamy, że w komórce za rozkazem znajduje się drugi argument rozkazu. Kolejny zaś rozkaz zapisany jest nie w kolejnej komórce za danym rozkazem, ale w w komórce następnej, tj. o adresie (L)+2.

Jeżeli Z=1

4. *wyak weja ode weak*;
5. *wyak wes wyad wea*;
6. *pisz wyl wea*;

10. Rozkaz powinien zrealizować odejmowanie od zawartości akumulatora podwojonej zawartości komórki pamięci, której adres podany jest argumentem rozkazu, o ile tylko zawartość tej komórki pamięci jest większa od wartości będącej przed wykonaniem rozkazu w akumulatorze. Do jego wykonania potrzebne są maksymalnie 4 fazy (takty zegara), w których muszą być aktywne odpowiednio sygnały sterujące:
    1. *czyt wys wei il*;
    2. *wyad wea*;
    3. *czyt wys weja ode weak*;

       Jeżeli Z=1
    4. *wys weja ode weak wyl wea*;

       w przeciwnym przypadku
    4.' *wys weja dod weak wyl wea*.

## Literatura


1. M. Chłopek, R. Tutajewicz, *Ćwiczenia laboratoryjne z Podstaw Informatyki – maszyna W*. Skrypt uczelniany Politechniki Śląskiej nr 2062. Wydawnictwo Politechniki Śląskiej, Gliwice, 1997.
2. H.H. Goldstine, *The Computer: from Pascal to von Neumann*. Princeton, New Jersey: Princeton University Pres, 1972.
3. K. Grochla, G. Hryń, S. Iwaszenko, P. Kasprzyk, J. Kubica, M. Widera, T. Wróbel, *Wykłady z Podstaw Informatyki profesora Stefana Węgrzyna*. Skrypt uczelniany Politechniki Śląskiej nr 2321. Wydawnictwo Politechniki Śląskiej, Gliwice, 2003.
4. L. Null, J. Lobur, *Struktura organizacyjna i architektura systemów komputerowych*. Helion, Gliwice, 2004.
5. D.A. Patterson, J.L. Hennessy, *Computer Organization and Design: The Hardware/Software Interface*, wydanie piąte, Elsevier Morgan Kaufmann, Oxford, 2014.
6. S. Węgrzyn, *Podstawy informatyki*. PWN, Warszawa, 1982.