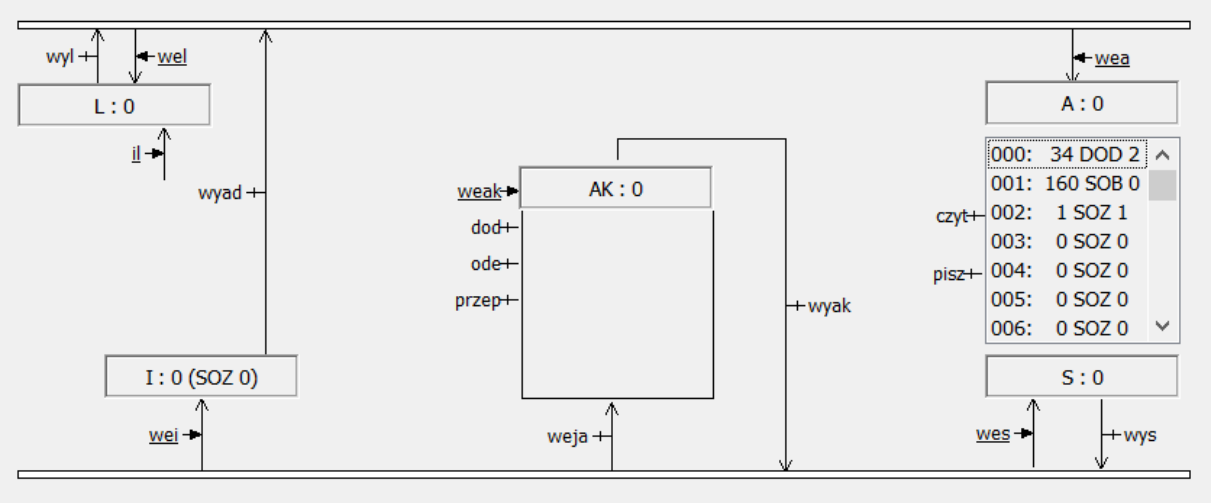
Maszyna W

Maszyna W to uproszczony model rzeczywistego komputera. Schemat najprostszej wersji maszyny W przedstawia rysunek [wstaw odwołanie].



Maszyna W składa się z kilku podstawowych elementów, takich jak pamięć operacyjna, jednostka arytmetyczno-logiczna i układ sterujący. Składowe te są połączone poprzez 2 magistrale: magistralę 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.

Pamięć operacyjna 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, każda komórka przechowuje pojedyncze słowo i jest identyfikowana poprzez numer komórki nazywany adresem. Aby wykonać jakąś operację na pamięci, procesor musi zawsze podać adres komórki, której ta operacja dotyczy. Aby odczytać określoną komórkę pamięci, należy najpierw wpisać adres komórki do rejestr adresowego pamięci. Sygnał *wea* 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*) i przesłanie odczytanej wartości na magistralę danych (sygnał *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. Sygnał *wes* 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.

Jednostka arytmetyczno-logiczna (JAL) jest elementem procesora odpowiedzialnym za wykonywanie obliczeń. Typowe operacje wykonywane w JAL to dodawanie, odejmowanie, suma i iloczyn logiczny oraz przesunięcia bitowe. Jednostka arytmetyczno-logiczna

w maszynie W ma tylko jedno wejście. Z tego powodu podczas wykonywania operacji dwuargumentowych przyjmuje się, że pierwszy argument znajduje się w związanym z JAL rejestrze, zwanym rejestrem akumulatora AK. Zanim zostanie wykonana właściwa operacja dwuargumentowa należy wartość pierwszego argumentu wpisać do akumulatora. W akumulatorze zapisywane są także wyniki obliczeń wykonywanych w JAL. W najprostszej wersji maszyny W, jednostka arytmetyczno-logiczna potrafi wykonywać tylko 3 operacje: przepisywanie stanu wejścia na wyjście (aktywowane sygnałem *przep*), dodawanie arytmetyczne (sygnał *dod*) i odejmowanie (sygnał *ode*). Sygnał *weja* powoduje udostępnienie wartości na magistrali na wejście JAL, zaś sygnał *weak* pozwala wynik obliczeń zapisać do rejestru akumulatora. Zawartość rejestru akumulatora można wyprowadzić na magistralę danych, zadanie to realizuje sygnał *wyak*. Obliczenia w JAL wykonywane są na liczbach binarnych zapisanych w zapisie uzupełnieniowym do 2. 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 1. W przeciwnym przypadku bit ten będzie równy 0. Układ sterujący procesora może sterować wykonywaniem obliczeń w różny sposób w zależności od stanu bitu Z.

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 tego rejestru. Ponadto z licznikiem rozkazów związane są 2 inne sygnały. Sygnał *wel* aktywuje operację zapisania w rejestrze L aktualnej zawartości magistrali adresowej, zaś sygnał *wyl* powoduje wyprowadzenie zawartości rejestru na magistralę. W rejestrze instrukcji przez większość czasu wykonywania rozkazu znajduje się słowo opisujące wykonywany rozkaz. Rejestr ten można podzielić na 2 części. Starsza część rejestru przechowuje kod rozkazu, zaś młodsza AD argument. Najczęściej argumentem jest adres komórki, do której odwołuje się dany rozkaz. Sygnał *wyad* pozwala wyprowadzić argument na magistralę danych.. Drugim sygnałem związanym z rejestrem instrukcji jest sygnał *wei*. Powoduje on przepisanie słowa opisującego rozkaz do wykonania z magistrali danych do rejestru I. Należy podkreślić, że w poprawnie zaprojektowanym rozkazie sygnał *wei* występuje zawsze tylko i wyłącznie w pierwszej fazie rozkazu.

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, powodujących 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). W kolejnych fazach będą realizowane dalsze etapy wykonania rozkazu. Są to: ustalenie adresu efektywnego argumentu, jego odczyt i wykonanie na nim właściwej operacji oraz przygotowanie do wykonania następnego rozkazu. Cały proces projektowania rozkazów zostanie przedstawiony na przykładzie rozkazu dodawania.

Przykład 1

Zaprojektować rozkaz dodawania DOD Ad. W wyniku działania tego rozkazu do akumulatora należy dodać zawartość komórki pamięci, której adres jest argumentem projektowanego rozkazu.

Działanie rozkazu można opisać za pomocą przesyłu międzyrejestrowego (Ak)+((Ad))→Ak. Podobnie jak w przypadku każdego innego 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* i *il*. Te właśnie sygnały będą tworzyć pierwszą fazę wykonania rozkazu dodawania.

Po zdekodowaniu konkretnego rozkazu można już przystąpić do charakterystycznych dla niego działań. W tym przypadku do akumulatora musimy dodać zawartość komórki pamięci, której adres jest argumentem rozkazu. Samo dodawanie będzie poprzedzone odczytaniem zawartości odpowiedniej komórki pamięci. 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. 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 (*dod*) i zapisanie jego wyniku w akumulatorze (*weak*). 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). Ten przesył zostanie zrealizowany dzięki sygnałom *wyl* i *wea*.

Tym samym końcową postać rozkazu dodawania tworzą trzy fazy. W pierwszej aktywne muszą być sygnały: *czyt*, *wys*, *wei* i *il*; w drugiej: *wyad* i *wea*; wreszcie w trzeciej: *czyt*, *wys*, *weja*, *dod*, *weak*, *wyl* i *wea*. Działanie zaprojektowanego rozkazu można przetestować dzięki programowemu symulatorowi maszyny W. Po jego uruchomieniu, należy zwykle utworzyć nowy plik z opisem rozkazu, skompilować go (co spowoduje dodanie rozkazu do listy rozkazów maszyny W) a następnie sprawdzić jego działanie[1]. Opis rozkazu dodawania zamieszczono poniżej:

```
ROZKAZ DOD;
czyt wys wei il;
wyad wea;
czyt wys weja dod weak wyl wea;
```

Pierwsza linia zawiera nazwę rozkazu, w kolejnych liniach wymieniono sygnały sterujące aktywne w poszczególnych taktach rozkazu. Średnik na końcu każdej linii (od 2 do 4) oznacza koniec taktu. Rozkaz DOD za każdym razem będzie wykonywany tak samo. Jednakże istnieją rozkazy, które realizowane są różnie w zależności od aktualnego stanu procesora.

Przykład 2

Zaprojektować rozkaz skoku warunkowego SOZ Ad, który spowoduje ustalenie adresu następnego rozkazu w zależności od zawartości akumulatora. Jeśli w akumulatorze jest liczba różna od 0, następnym wykonywanym rozkazem ma być rozkaz znajdujący się w kolejnej komórce pamięci. Jeżeli zawartość akumulatora jest równa 0, następnym rozkazem będzie ten, który zapisano w komórce pamięci o adresie podanym jako argument rozkazu.

Głównym celem rozkazu SOZ jest ustalenie adresu następnego wykonywanego rozkazu. Jeśli w akumulatorze będzie 0, jako następny zostanie wykonany rozkaz, którego adres podano jako argument rozkazu SOZ. W przeciwnym razie przejdziemy do wykonywania rozkazu znajdującego się w pamięci bezpośrednio za rozkazem SOZ.

Jak zwykle pierwszą fazę projektowanego rozkazu tworzą sygnały czyt, wys, wei i il. Następnie należy sprawdzić, czy w akumulatorze znajduje się 0. Jest to możliwe dzięki istnieniu odpowiednich sygnałów stanu procesora. W maszynie W mamy do dyspozycji 2 takie sygnały: Z (znak liczby) – informujący czy w akumulatorze jest liczba ujemna oraz ZAK, który przyjmuje wartość 1, gdy w akumulatorze jest 0. Aby zaprojektować rozkaz SOZ należy skorzystać z sygnału ZAK. Jeżeli ZAK = 1, to w akumulatorze jest 0 a to oznacza, że należy wykonać skok, czyli przesłać do rejestrów L i A adres znajdujący się w części adresowej rejestru instrukcji. Wymaga to uaktywnienia sygnałów *wyad*, *wel* i *wea*. Jeżeli ZAK = 0, to w akumulatorze jest liczba różna od 0 a w takim przypadku adres następnego rozkazu znajduje się w rejestrze L (po wykonaniu inkrementacji w pierwszej fazie) i wystarczy go przesłać do rejestru adresowego pamięci. Z tym przesyłem związane są sygnały *wyl* i *wea*. Pełny projekt rozkazu SOZ zamieszczono poniżej.

```
ROZKAZ SOZ;
czyt wys wei il;
JEZELI ZAK TO @zero GDY NIE @niezero;
```

```
@zero wyad wea wel KONIEC;
@niezero wyl wea;
```

Warto zwrócić uwagę na trzecią linię kodu. Zawiera ona sprawdzenie stanu testowanego sygnału ZAK. Jeżeli ZAK = 1 (jest 0 w Ak) następuje przejście do wykonywania sygnałów zapisanych w linii oznaczonej etykietą @zero. Gdy ZAK = 0, jako następne zostaną wykonane sygnały sterujące wpisane w linii której etykieta (@niezero) pojawia się po frazie GDY NIE. Zauważmy też, że w sytuacji, gdy podawana jest lista sygnałów aktywnych w ostatnim takcie rozkazu a nie jest to ostatnia linia opisu rozkazu, konieczne jest dopisanie słowa kluczowego KONIEC (tak jak jest to w czwartej linii opisu rozkazu SOZ).

Przykład 3

Zaprojektować rozkaz dekrementacji akumulatora DEC. W wyniku działania tego rozkazu od akumulatora należy odjąć jedynkę. Wykorzystać architekturę W+.

Zadanie to to przykład bardziej złożonego rozkazu, wymagającego rozbudowy maszyny W o dodatkowe połączenie, umożliwiające przesył danych bezpośrednio między magistralami słowową i adresową. Jego rozwiązanie zostanie przedstawione za pomocą listy elementarnych przesyłów międzyrejestrowych. Następnie do każdego przesyłu zostaną przyporządkowane odpowiednie sygnały sterujące i wreszcie na koniec sygnały te zostaną odpowiednio rozmieszczone w czasie (tzw. podział na takty). Jest to typowa kolejność działań przy projektowaniu rozkazów.

Niestety nasza jednostka arytmetyczno-logiczna jest bardzo uboga i nie oferuje bezpośrednio operacji inkrementacji i dekrementacji akumulatora. Stąd konieczne jest bardziej skomplikowane postępowanie. Najpierw do akumulatora zostanie dodana aktualna zawartość licznika rozkazów, następnie licznik zostanie zinkrementowany (zostanie dodana jedynka do zawartości licznika). Z kolei zwiększona zawartość licznika zostanie odjęta od akumulatora, co w efekcie spowoduje, że w akumulatorze znajdzie się wartość o 1 mniejsza od jego pierwotnej zawartości. Oczywiście nie wolno zapomnieć o przywróceniu oryginalnej zawartości licznika rozkazów. Będzie to możliwe dzięki wcześniejszemu jej zapamiętaniu w rejestrze S. Całość działań składających się na wykonanie rozkazu dekrementacji przedstawiają poniższe przesyły międzyrejestrowe.

((A)) → I
(L) + 1 → L
(L) → S
(Ak) + (L) → Ak
(L) + 1 → L
(Ak) – (L) → Ak
(S) → L
(S) → A

Z każdym z powyższych przesyłów wiążą się określone sygnały sterujące. Dopiszmy je zatem obok związanych z nimi przesyłów. Dodatkowo poszczególne linie zostaną ponumerowane.

1. ((A)) → I *czyt wys wei*

2. (L) + 1 → L *il*
3. (L) → S *wyl as wes*
4. (Ak) + (L) → Ak *wyl as weja dod weak*
5. (L) + 1 → L *il*
6. (Ak) – (L) → Ak *wyl as weja ode weak*
7. (S) → L *wys sa wel*
8. (S) → A *wys sa wea*

Sygnały *as* i *sa* są związane w wprowadzonym połączeniem między magistralami. Sygnał *as* powoduje przesłanie na magistralę danych zawartości magistrali adresowej, zaś sygnał *sa* przepisze zawartość magistrali danych na magistralę adresową („zgubione" zostaną najstarsze bity stanu magistrali danych).

Aby uzyskać pełny projekt rozkazu musimy jeszcze rozmieścić poszczególne operacje w czasie. Dążymy przy tym do wykonania możliwie wielu operacji w pojedynczym takcie.

Tradycyjnie przesyły przedstawione w dwóch pierwszych liniach realizowane są w pierwszej fazie każdego rozkazu. W kolejnej fazie zostaną wykonane przesyły z linii 3, 4 i 5. Trzecią fazę będzie stanowić przesył umieszczony w linii 6. Zaś dwa ostatnie można połączyć jako ostatnią fazę wykonania rozkazu. Tym samym ostateczny projekt rozkazu będzie mieć postać jak poniżej

```
ROZKAZ DEC;
ARGUMENTY 0;
czyt wys wei il;
wyl as wes weja dod weak il;
wyl as weja ode weak;
wys sa wel wea;
```

Wyjaśnienia wymaga polecenie umieszczone w drugiej linii opisu rozkazu. Projektowany przez nas rozkaz nie wykorzystuje pola przeznaczonego na argument (dekrementowana jest zawartość akumulatora a nie np. jakiejś komórki w pamięci, której adres mógłby być argumentem rozkazu). Ten fakt został zaznaczony odpowiednim wpisem.

Zadania do wykonania

Przesyły międzyrejestrowe

Posługując się mechanizmem sterowania ręcznego zrealizować następujące przesyły międzyrejestrowe (nie rozkazy !). Przyjąć, że zawartości poszczególnych rejestrów nie są określone (odpowiednie wartości liczbowe należy „wypracować")

1. ((AD)) → Ak
2. (Ak) shl 1 → 0 // przesunięcie logiczne o 1 bit w lewo (maszyna W+)
3. |((L))| → Ak
4. |(Ak)| → (2)
5. (Ak) – ((L)) → Ak
6. (Ad) – (Ak) → 1
7. 2 * (Ak) → Ak

8. (0) + (1) → (Ad)
9. (Ak) → (L)
10. ((Ad)+1) – ((Ad)) → Ak
11. – (AK) → (Ad)
12. (Ak)+((L)) → (L)+1
13. (Ak) + (S) → (Ad)
14. ((Ad)) + (Ak) → (L)+1
15. |(Ak)| → (L)
16. ((Ad)) + ((Ad)+1) → Ak
17.

Rozkazy

1. Zaprojektować rozkaz MIN Ad porównujący zawartość akumulatora z zawartością komórki pamięci, której adres jest argumentem rozkazu i zapisujący w akumulatorze mniejszą z nich.
2. Zaprojektować rozkaz MAX Ad porównujący zawartość akumulatora z zawartością komórki pamięci, której adres jest argumentem rozkazu i zapisujący w akumulatorze większą z nich.
3. Zaprojektować rozkaz MN2 Ad zapisujący w akumulatorze 2-krotność zawartości komórki pamięci, której adres jest argumentem rozkazu.
4. Zaprojektować rozkaz ABS Ad wpisujący do akumulatora wartość bezwzględną liczby znajdującej się w komórce pamięci, której adres jest argumentem rozkazu
5. Zaprojektować rozkaz skoku warunkowego SNZ Ad, który spowoduje ustalenie adresu następnego rozkazu w zależności od zawartości akumulatora. Jeśli w akumulatorze jest 0, następnym wykonywanym rozkazem ma być rozkaz znajdujący się w kolejnej komórce pamięci. Jeżeli zawartość akumulatora jest różna od 0, następnym rozkazem będzie ten który zapisano w komórce pamięci o adresie podanym jako argument rozkazu.
6. Zaprojektować bezargumentowy rozkaz dekrementacji akumulatora. W wyniku działania tego rozkazu zawartość akumulatora powinna zostać zmniejszona o 1.
7. Zaprojektować rozkaz dodawania z adresacją pośrednią DDP Ad. W wyniku działania tego rozkazu do akumulatora należy dodać zawartość komórki pamięci, której adres znajduje się w komórce pamięci, której adres jest argumentem rozkazu. Wykorzystać architekturę W+.
8. Zaprojektować rozkaz odejmowania z adresacją pośrednią ODP Ad. W wyniku działania tego rozkazu od akumulatora należy odjąć zawartość komórki pamięci, której adres znajduje się w komórce pamięci, której adres jest argumentem rozkazu. Wykorzystać architekturę W+.
9. Zaprojektować rozkaz pobierania z adresacją pośrednią PBP Ad. W wyniku działania tego rozkazu do akumulatora należy przesłać zawartość komórki pamięci, której adres znajduje się w komórce pamięci, której adres jest argumentem rozkazu. Wykorzystać architekturę W+.
10. Zaprojektować rozkaz ładowania do pamięci z adresacją pośrednią ŁDP Ad. W wyniku działania tego rozkazu zawartość akumulatora należy zapisać w komórce pamięci, której adres znajduje się w komórce pamięci, której adres jest argumentem rozkazu. Wykorzystać architekturę W+.
11. Zaprojektować rozkaz zamiany zawartości 2 komórek pamięci: komórki, której adres jest umieszczony w części adresowej rozkazu i komórki znajdującej się bezpośrednio

za rozkazem. Przyjąć, że następny rozkaz znajduje się o 2 komórki za projektowanym rozkazem.