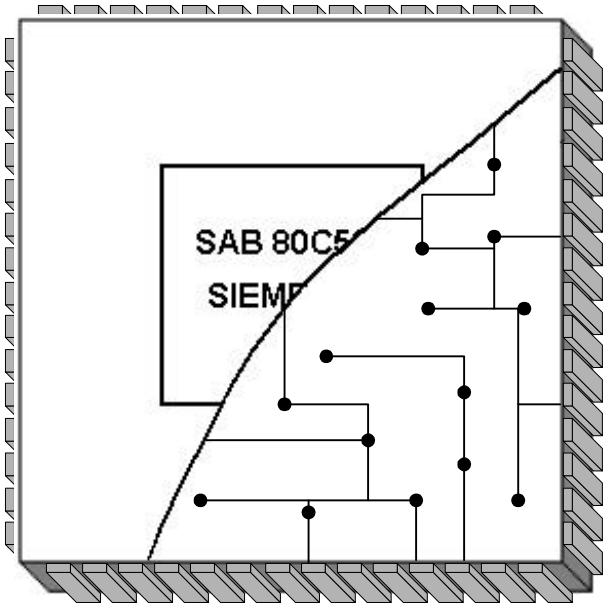
KRZYSZTOF P. DYRCZ CZESLAW T. KOWALSKI

ZDZISLAW ZARCZYNSKI

Podstawy techniki mikroprocesorowej



SPIS TRESCI

PRZEDMOWA

Opanowanie w latach siedemdziesiatych i osiemdziesiatych masowej i taniej produkcji mikroprocesorów stanowilo epokowe osiagniecie, zmieniajace zycie i otoczenie czlowieka. Mozliwe stalo sie wprowadzenie, w zastosowaniach przemyslowych i innych, cyfrowych metod sterowania i regulacji, gwarantujacych duza dokladnosc i pewnosc dzialania oraz znaczne zwiekszenie zakresu mozliwosci funkcjonalnych róznych urzadzen.

Wspólczesne uklady sterowania sa budowane najczesciej przy uzyciu mikroprocesorów. Dzieki nim mozliwa staje sie optymalizacja wlasciwosci dynamicznych sterowanych urzadzen przez wprowadzenie coraz bardziej skomplikowanych i rozbudowanych algorytmów, z jednoczesnym dopasowywaniem funkcji sterowania do róznych wymagan uzytkownika i wlasciwosci obiektu. Równoczesnie usprawnienie i przyspieszenie inzynierskich dzialan projektowych, uruchomieniowych i naprawczych wymaga poslugiwania sie systemami mikroprocesorowymi wyposazonymi w bogate oprogramowanie wspomagajace. Podane fakty uzasadniaja wprowadzenie do programu studiów podstawowych na Wydziale Elektrycznym przedmiotu *Podstawy techniki mikroprocesorowej*.

Niniejsza praca jest adresowana do studentów studiów inzynierskomagisterskich na kierunku Elektrotechnika, jako material pomocniczy do wykladów i cwiczen laboratoryjnych (w wymiarze 1W2L). Wyjasnia podstawowe problemy zwiazane z dzialaniem i programowaniem 8–bitowych mikroprocesorów. Przedstawiono budowe i programowanie mikrokomputera jednoukladowego **SAB 80C535**, ze wzgledu na duza jego popularnosc w zastosowaniach przemyslowych w Europie. Oprócz omówienia struktury sprzetowej mikrokomputera, mechanizmów przeplywu informacji oraz zasad programowania, zamieszczono liczne przyklady programów.

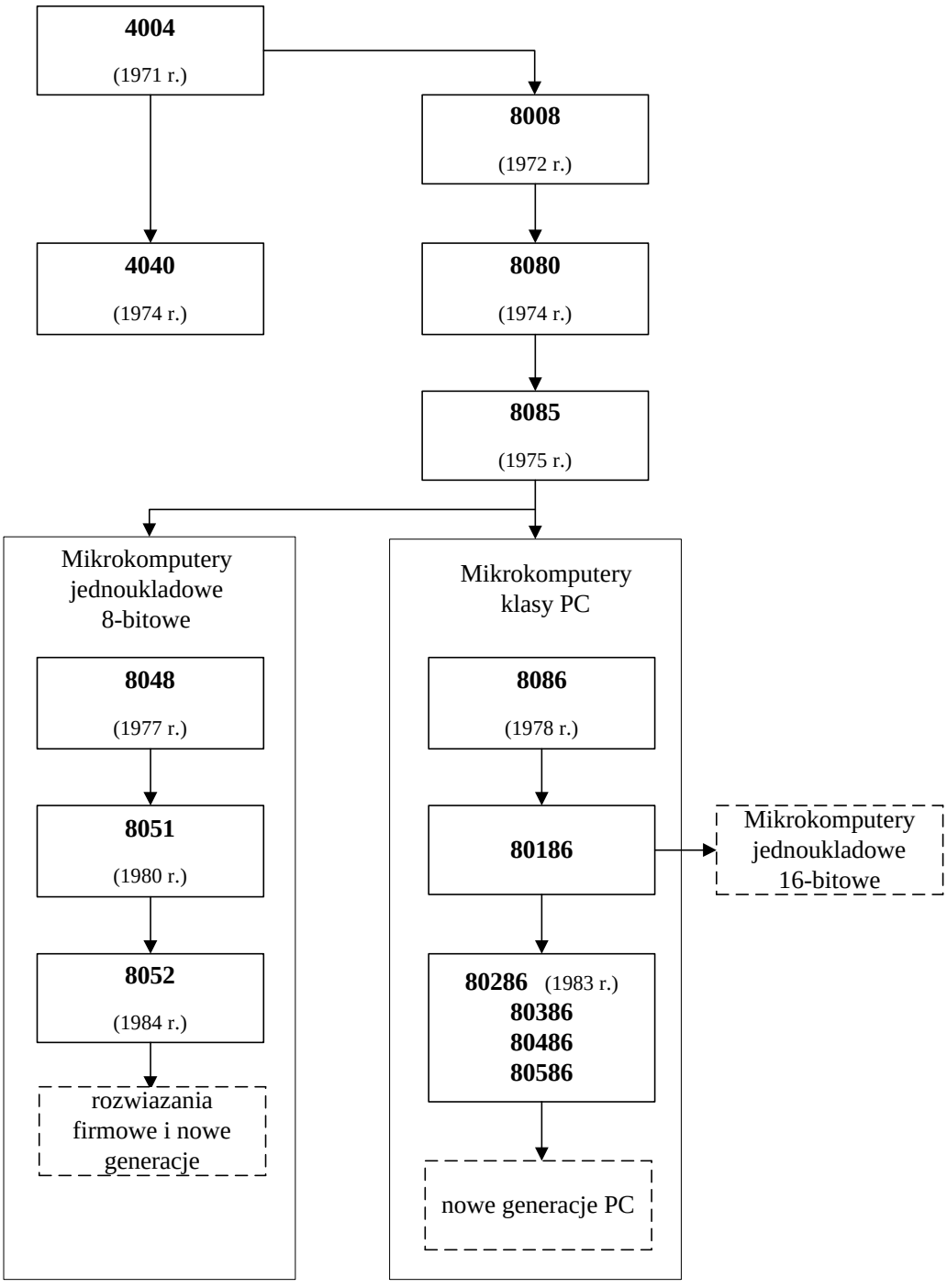
1. WIADOMOSCI OGÓLNE O MIKROPROCESORACH

1.1. Wprowadzenie

Historia powstania pierwszego ukladu wielkiej skali integracji, zwanego mikroprocesorem, rozpoczela sie w roku 1969. Wlasnie wtedy japonska firma Busicom, produkujaca kalkulatory, zamówila w amerykanskiej firmie INTEL zestaw mikroukladów, które mialy sluzyc do skladania kalkulatorów o róznych mozliwosciach. Firma INTEL miala juz za soba wiele doswiadczen zebranych podczas produkcji scalonych ukladów, zwlaszcza pamieci dostarczanych producentom systemów komputerowych. Efektem dzialalnosci specjalistów firmy INTEL, pod kierownictwem Marciana E. Hoffa, bylo skonstruowanie nie tylko zamówionego zestawu ukladów, ale równiez w 1971 roku pierwszego mikroprocesora, oznaczonego symbolem 4004, skladajacego sie z okolo 2300 tranzystorów.

Od momentu pojawienia sie pierwszego mikroprocesora rozpoczal sie prawdziwy wyscig, trwajacy do dzisiaj, polegajacy na konstruowaniu ukladów mikroprocesorowych coraz szybszych i o coraz wiekszych mocach obliczeniowych. Mikroprocesor 4004 byl 4–bitowy, co oznaczalo, ze informacje wewnatrz i na zewnatrz ukladu byly przetwarzane i przesylane 4–bitowymi porcjami. Stanowilo to wielkie ograniczenie dla szybkosci dzialania mikroprocesora, dlatego naturalnym dazeniem konstruktorów bylo rozszerzenie jednorazowo przesylanej informacji najpierw do 8–, nastepnie 16–, 32– a ostatnio 64– i 128–bitów. Mikroprocesory 64– i 128–bitowe sa konstrukcjami niezwykle zaawansowanymi technicznie. Wyprodukowany w roku1972 mikroprocesor 8–bitowy byl pierwszym przedstawicielem tzw. „rodziny 8080”, która stala sie swiatowym standardem. Na rysunku 1.1 przedstawiono poczatek „drzewa genealogicznego” wspólczesnych mikroprocesorów firmy INTEL. W roku 1976 pojawila sie trzecia generacja mikroprocesorów, produkowanych przez rózne firmy, charakteryzujaca sie zwiekszonym repertuarem rozkazów i trybów adresowania pamieci, wykonywaniem operacji na danych 8– i 16–bitowych, coraz wieksza szybkoscia dzialania. Obejmowala ona trzy klasy ukladów:

– mikroprocesory 8–bitowe (np.:Z80 firmy Zilog oraz 6809 firmy Motorola),
– mikroprocesory 16–bitowe (np.: 8086 firmy INTEL, Z8000 firmy Zilog oraz 68000 firmy Motorola),
– mikrokomputery jednoukladowe 8–bitowe (np.:8051 firmy INTEL, Z8 firmy Zilog oraz 6801 firmy Motorola).



Rys. 1.1. „Drzewo genealogiczne” mikroprocesorów firmy INTEL

Mikroprocesory 8– i 16–bitowe stanowily podstawe wielu konstrukcji opracowanych przez rózne firmy i stosowanych coraz powszechniej w przemysle. Dalszy rozwój konstrukcji mikroprocesorów (80286, 80386, 80486, 80586...) zostal zdominowany przez zastosowania w komputerach ogólnego przeznaczenia. Do bezposredniego natomiast sterowania cyfrowego urzadzeniami przemyslowymi byly i sa rozwijane mikroprocesorowe sterowniki jednoukladowe. Do grupy tej naleza m. in. uklady scalone rodziny INTEL MCS-48 i MCS-51. Rodziny te obejmuja kilka sterowników 8–bitowych rózniacych sie rozmiarami bloków pamieci. Grupa mikroprocesorowych sterowników jednoukladowych szybko powieksza sie o coraz to nowe rodziny. Obserwuje sie przy tym tendencje do róznicowania budowy ukladów z mysla o lepszym przystosowaniu ich do sterowania róznych grup obiektów.

Jak dotychczas na rynku europejskim dominuja mikrokontrolery jednoukladowe 8–bitowe wywodzace sie z rodziny INTEL 8051. Dlatego tez ten typ mikroprocesora bedzie podstawa dalszych rozwazan.

1.2. Podstawowe pojecia

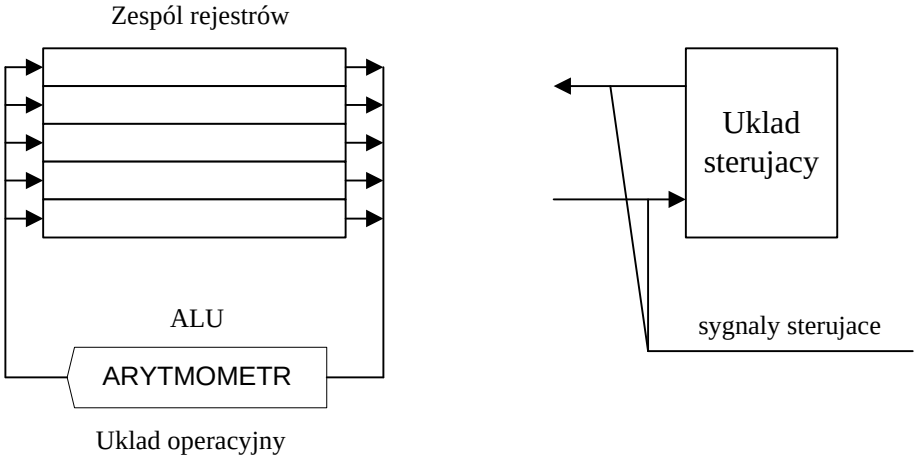
Mikroprocesor (ang. *microprocessor*) jest to procesor wykonany w postaci pojedynczego mikroukladu (ang. *Chip*) o wielkim stopniu scalenia. Okreslenie mikroprocesora zawiera w sobie dwa elementy:

- funkcjonalne przeznaczenie do przetwarzania informacji w zadany z zewnatrz (przez uzytkownika) sposób (procesor),
- technologiczne wykonanie jako elektronicznego ukladu scalonego o wielkim stopniu integracji.

Mikroprocesor, jak kazdy procesor, jest urzadzeniem przeznaczonym do bardzo szybkiego wykonywania dowolnego ciagu prostych operacji wybieranych sposród ustalonego zbioru operacji podstawowych (arytmetycznych i logicznych).

Zasadniczymi ukladami mikroprocesora sa: *arytmometr*, zwany tez *jednostka arytmetyczno-logiczna* (ang. *Arithmetic-Logic Unit ALU*), *zespól rejestrów uniwersalnych* (ang. *register bank*) oraz *uklad sterujacy* (ang. *control circuit*). Na rysunku 1.2

przedstawiono schematycznie zasadnicze zespoly struktury mikroprocesora (procesora).



Rys. 1.2. Podstawowe elementy struktury mikroprocesora

Kazdy rejestr stanowi komórke pamieci, w której moze byc przechowywane jedno slowo maszynowe. W czasie pracy procesora rejestry sa uzywane do chwilowego przechowywania danych i wyników wykonywanych operacji. Przetwarzanie danych zapisanych w rejestrach wewnetrznych procesora lub komórkach pamieci jest realizowane przez arytmometr ALU. Zespól rejestrów i arytmometrów tworzy tzw. uklad operacyjny procesora.

Przeplyw danych i wyników miedzy rejestrami, pamiecia i arytmometrami jest sterowany przez uklad sterujacy, który decyduje o tym, jakie operacje i na jakich danych maje byc wykonane. Uklad sterujacy jest odpowiedzialny za pobieranie kolejnych rozkazów programu z pamieci, interpretowanie tresci pobranych rozkazów i ich wykonanie w ukladzie operacyjnym. Dzialanie procesora jest cykliczne i w kolejnych cyklach pracy (cyklach rozkazowych) z komórki pamieci sa pobierane kolejne rozkazy programu. Pobrany rozkaz jest przesylany do ukladu sterujacego, który rozpoznaje typ rozkazu i ustala rodzaj operacji, która ma byc wykonana i identyfikuje argumenty operacji. Nastepnie steruje pobraniem odpowiednich danych i wykonaniem operacji w ukladzie operacyjnym. Dane stanowiace argumenty operacji sa pobierane z rejestrów procesora lub z komórki pamieci. Pobranie danych z rejestrów (a nie z pamieci) pozwala na szybsze wykonanie operacji. Liczba rejestrów uniwersalnych procesora jest

zwykle ograniczona. Zarówno rozkazy, jak i dane sa zapisywane w identycznej postaci slów zerojedynkowych. O tym, jak procesor zidentyfikuje slowo pobrane z pamieci decyduje miejsce, do którego slowo to zostanie przeslane wewnatrz procesora. Slowa przeslane do ukladu sterujacego sa traktowane jako rozkazy, slowa przeslane do ukladu operacyjnego sa przetwarzane jako dane. *Slowem* nazywa sie wektor informacji cyfrowej, który moze mikroprocesor wymienic z pamiecia w wyniku jednej operacji czytania lub zapisu. *Dlugosc slowa* (liczba bitów) mikroprocesora jest jednym z jego najwazniejszych parametrów. Determinuje ona w decydujacym stopniu jego efektywnosc (moc obliczeniowa); wieksza dlugosc slowa oznacza wieksza róznorodnosc rozkazów i mozliwosc stosowania róznych technik adresowania, ale z drugiej strony istotnie zwieksza zlozonosc budowy i podnosi cene. Liczba bitów w slowie mikroprocesora jest równa liczbie linii jego magistrali danych. Z reguly nie jest ona równa liczbie linii jego wewnetrznej magistrali.

W róznych mikroprocesorach stosuje sie rózna dlugosc slowa; zwykle jednak jest to wielokrotnosc 8 bitów (8 bitów = 1 bajt).

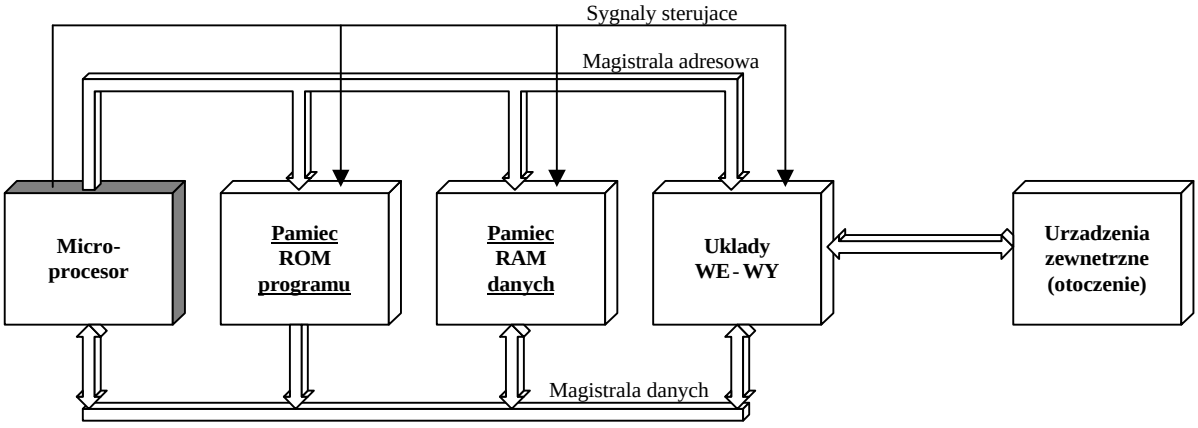
Sam mikroprocesor nie jest zdolny do samodzielnego funkcjonowania. Do jego prawidlowej pracy potrzebne sa dwa typy ukladów dodatkowych:

- uklady do wprowadzania i wyprowadzania informacji, nazywane *ukladami wejscia-wyjscia* (ang. *input-output*),
- *pamiec* (ang. *memory*), w której jest przechowywany program oraz dane i wyniki obliczen, zarówno koncowe, jak i czastkowe.

Mikroprocesor jako *jednostka centralna* (ang. *Central Processing Unit CPU*) wraz z zestawem ukladów dodatkowych tworza *system mikroprocesorowy* zwany równiez mikrokomputerem (komputerem). Na rysunku 1.3. przedstawiono schemat ideowy struktury mikrokomputera.

Rozwój technologii umozliwiajacy scalanie coraz wiekszych ukladów doprowadzil do zbudowania równiez calych mikrokomputerów zawartych w jednym mikroukladzie. Sa to tzw. *mikrokomputery jednoukladowe* (ang. *single chip microcomputer*) w postaci ukladu scalonego zawierajacego w sobie wszystkie bloki funkcjonalne (jednostke centralna, pamiec, uklady wejscia–wyjscia) tworzace mikrokomputer zdolny do

samodzielnego dzialania.



Rys.1.3. Schemat ideowy struktury mikrokomputera

Mikrokomputery jednoukladowe maja wiele zalet w stosunku do mikrokomputerów wieloukladowych, zwlaszcza w zastosowaniach wymagajacych malych systemów mikroprocesorowych, np. w przemyslowych ukladach automatyki, urzadzeniach powszechnego uzytku itp. Te zalety to:

– wysoka niezawodnosc wynikajaca z mniejszej liczby elementów w ukladzie,
– male wymiary systemu,
– niskie koszty projektowania oraz realizacji systemów uzytkowych.

W duzych systemach, wymagajacych duzych pojemnosci pamieci i zlozonych urzadzen zewnetrznych, wymienione zalety traca znaczenie i stosowanie w nich mikrokomputerów jednoukladowych jest nieuzasadnione, a czesto niemozliwe. Obecnie coraz powszechniej mikrokomputery jednoukladowe nazywa sie *mikroprocesorowymi sterownikami jednoukladowymi* lub *mikrokontrolerami*, ze wzgledu na projektowanie ich pod katem realizacji zlozonych funkcji sterujacych w zastosowaniach przemyslowych.

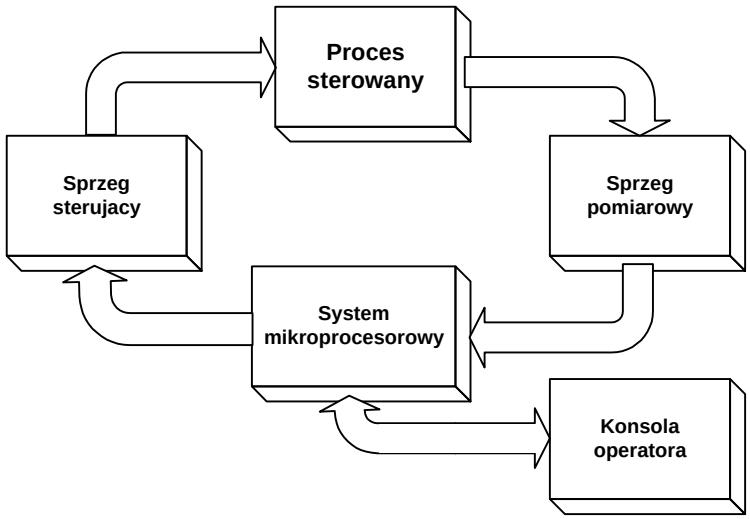
Rodzina mikrokontrolerów szybko powieksza sie o nowe rozwiazania, obserwuje sie przy tym tendencje do róznicowania budowy ukladów z mysla o lepszym przystosowaniu do sterowania róznych typowych grup obiektów technicznych.

Ze wzgledu na zapotrzebowanie na przetwarzanie cyfrowe sygnalów analogowych szybkozmiennych pojawily sie tzw. *mikroprocesory sygnalowe* (ang. *Digital Signal Processors DSP*). Mikroprocesory sygnalowe umozliwiaja realizacje zlozonych

algorytmów obliczeniowych (sterujacych) w bardzo krótkim czasie. Zastosowanie tak szybkich procesorów w ukladach sterowania otworzylo bardzo obiecujaca perspektywe realizacji algorytmów znanych z teorii sterowania, a dotychczas niezrealizowanych technicznie.

W dalszych rozwazaniach uwaga zostanie skupiona przede wszystkim na mikroprocesorowych sterownikach jednoukladowych, czyli mikrokontrolerach, ze wzgledu na ich aktualna role w zastosowaniach przemyslowych.

System mikroprocesorowy komunikuje sie z realnym procesem sterowania za posrednictwem urzadzen laczacych, noszacych nazwe *sprzegów* (interfejsów, adapterów). Na rysunku 1.4 przedstawiono drogi przesylania sygnalów miedzy procesem a systemem mikroprocesorowym.



Rys. 1.4. Sposób wlaczenia systemu mikroprocesorowego w proces sterowany

Sprzegi pomiarowe przetwarzaja sygnaly naplywajace z czujników pomiarowych, zadajników itp. W sklad tych sprzegów moze równiez wchodzic przetwornik analogowo–cyfrowy. Zadaniem *sprzegów sterujacych* jest, oprócz przetworzenia postaci sygnalów, zapewnienie odpowiedniej mocy sygnalów wyjsciowych. W przypadku stosowania systemów mikroprocesorowych ogólnego przeznaczenia, uzytkownik zwykle musi projektowac indywidualnie poszczególne sprzegi pomiarowe i sterujace. W przypadku mikrokontrolerów natomiast coraz czesciej uzytkownik ma juz ulatwione zadanie, gdyz sa one standardowo wyposazone w podstawowe uklady sprzegajace, tzn.

w przetworniki analogowo–cyfrowe i cyfrowo–analogowe, wejscia i wyjscia cyfrowe, uklady licznikowe. Dzieki takim rozwiazaniom znacznie upraszcza sie konstrukcja przemyslowych sterowników mikroprocesorowych.

1.3. Budowa i dzialanie mikroprocesora

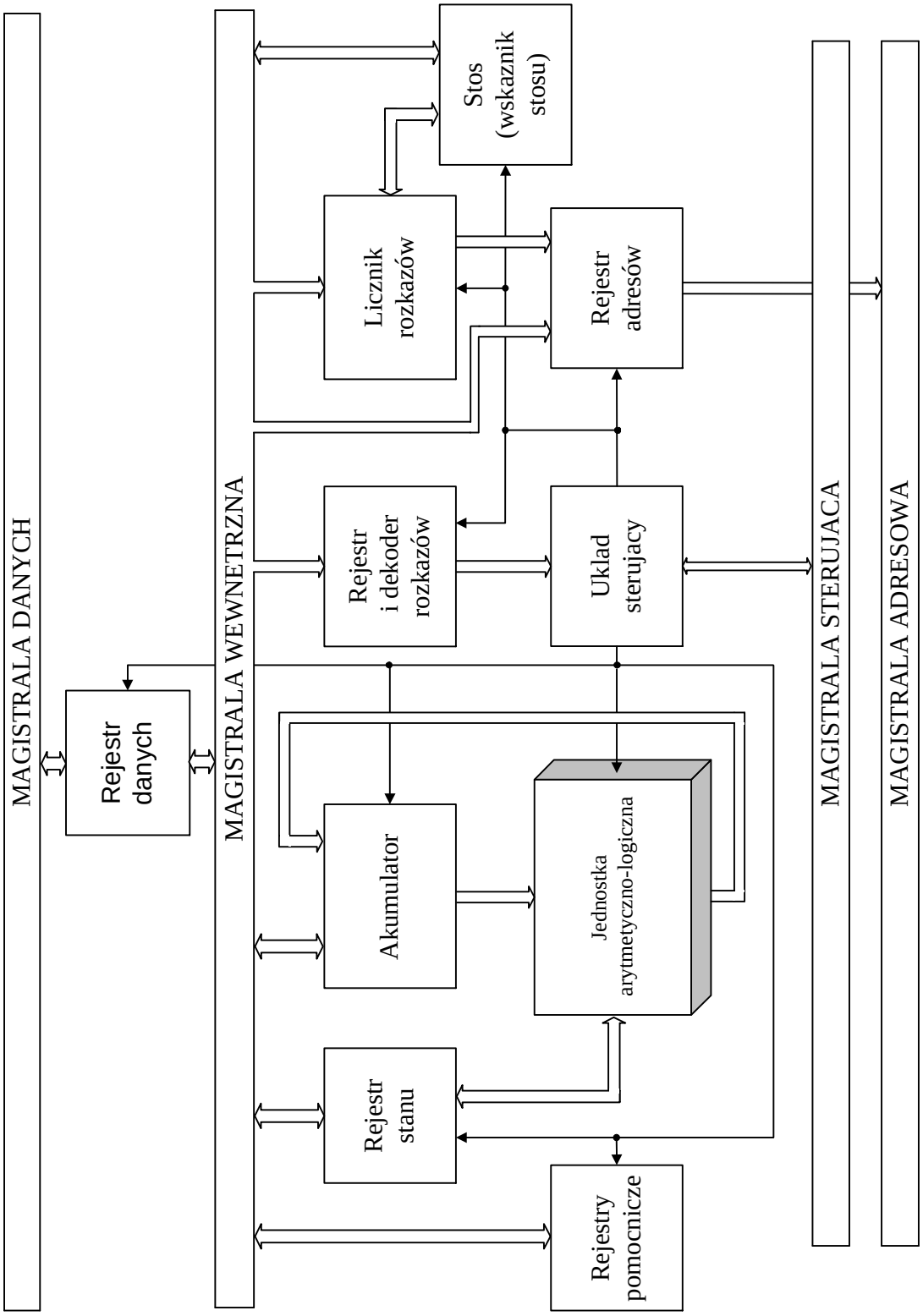
Jak przedstawiono w rozdziale 1.2, mikroprocesor moze funkcjonowac wylacznie jako element systemu mikroprocesorowego. W budowie mikroprocesorów wytwarzanych przez róznych producentów wystepuja znaczne róznice. Jednakze w kazdym z nich mozna wyróznic wspólne bloki funkcjonalne. Dlatego celowe jest zapoznanie sie z ogólna budowa i dzialaniem podstawowych ukladów mikroprocesora, gdyz ulatwi to zrozumienie budowy i dzialania konkretnego rozwiazania sprzetowego.

Na rysunku 1.5 przedstawiono podstawowe uklady mikroprocesora. Uklady te wchodza w sklad trzech zespolów funkcjonalnych:

– jednostki arytmetyczno-logicznej ALU,
– ukladu sterowania wraz z rejestrem rozkazów i dekoderem rozkazów,
– zespolu rejestrów jednostki centralnej (m.in. akumulator, rejestr stanu, rejestr danych, rejestr adresów, wskaznik stosu, rejestry pomocnicze).

Zespoly sa polaczone ze soba wewnetrzna szyna (magistrala) danych. Komunikacja z ukladami wspólpracujacymi (pamiec, uklady wejscia–wyjscia) prowadzona jest przez:

– magistrale adresowa, po której mikroprocesor wysyla adres pamieci lub urzadzenia wejscia–wyjscia,
– magistrale danych, po której sa przesylane informacje (kody rozkazów i liczby) miedzy jednostka centralna a ukladami otoczenia,
– magistrale sterujaca, po której przez mikroprocesor sa przesylane sygnaly okreslajace rodzaj operacji, jaka ma wykonac uklad wspólpracujacy (np. odczyt lub zapis pamieci).



Rys. 1.5. Podstawowe uklady mikroprocesora

Jednostka arytmetyczno-logiczna realizuje podstawowe operacje arytmetyczne i logiczne na liczbach binarnych.

Rejestr i dekoder rozkazów wraz z *ukladem sterujacym* zarzadzaja praca calej struktury. Przez generacje odpowiednich wewnetrznych sygnalów sterujacych uklad sterowania:

- okresla rodzaj operacji wykonywanych przez ALU,
- steruje praca wszystkich rejestrów mikroprocesora,
- steruje przesylaniem informacji po wewnetrznej szynie danych.

Uklad sterowania wysyla równiez sygnaly sterujace do elementów zewnetrznych (pamieci i ukladów wejscia–wyjscia) okreslajace rodzaj operacji jaka maja one wykonac. W sklad ukladu sterowania wchodzi równiez rejestr rozkazów i dekoder rozkazów. Do rejestru rozkazów jest wpisywany pobrany z pamieci kod rozkazów, który jest dekodowany przez dekoder, po czym uklad sterowania wytwarza odpowiednie sygnaly sterujace konieczne do wykonania rozkazu. Sygnaly sterujace sa generowane w rytmie narzuconym przez czestotliwosc podstawowego zegara taktujacego. Mikroprocesor zawiera pewna liczbe rejestrów niedostepnych programowo dla uzytkownika i wykorzystywanych tylko przez uklad sterowania. Z punktu widzenia uzytkownika istotne sa tylko te rejestry mikroprocesora, których zawartosc moze byc przez niego odczytana lub zmieniona (za pomoca odpowiednich rozkazów) i które moze on wykorzystywac do pamietania wyników operacji lub sterowania programem. Sa to nastepujace rejestry:

- akumulator,
- zespól rejestrów uniwersalnych (pomocniczych),
- zespól rejestrów specjalnych (licznik rozkazów, wskaznik stosu, rejestry stanu i adresów).

Rejestry uniwersalne (pomocnicze) sa wykorzystywane przede wszystkim jako pamiec podreczna do przechowywania czesciowych wyników obliczen, stalych itp. Kazdy z nich ma dlugosc jednego slowa. Lista rozkazów obejmuje zwykle rozkazy umozliwiajace modyfikacje zawartosci przynajmniej czesci rejestrów pomocniczych. Ponadto rejestry bywaja polaczone w pary, co umozliwia operacje na slowach dwubaj-

towych. Grupa rejestrów pomocniczych pelni role podobna do notatnika umieszczonego w pamieci zapisywalnej, jednak korzystanie z nich znacznie ulatwia programowa realizacje algorytmu i skraca czas wykonania programu.

Akumulator jest jednym z najwazniejszych dla uzytkownika rejestrów mikroprocesora. Spelnia on takie funkcje, jak rejestry uniwersalne. Ponadto w akumulatorze jest zwykle zawarty jeden z argumentów operacji oraz zapisane wyniki operacji arytmetycznych i logicznych wykonywanych przez mikroprocesor. Dlugosc (liczba bitów) akumulatora i rejestrów uniwersalnych wyznacza podstawowy parametr mikroprocesora – dlugosc slowa. W grupie rejestrów specjalnych kazdy z nich spelnia swoja scisle okreslona funkcje i z reguly nie moze byc wykorzystywany w innych celach. *Licznik rozkazów*, zwany tez licznikiem kroków programu, jest rejestrem równoleglym, w którym jest formowany adres kolejnego rozkazu, jaki bedzie pobrany przez mikroprocesor z pamieci stalej. Po kazdym pobraniu kodu rozkazów zawartosc licznika rozkazów jest zwiekszana o 1. Dlugosc licznika rozkazów wyznacza zwykle maksymalna pojemnosc pamieci, jaka mozna bezposrednio dolaczyc do mikroprocesora. Przy dlugosci licznika równej 16 bitów maksymalna pojemnosc pamieci wyniesie 2^{16}=65535 bitów. Adresy pamieci o maksymalnej pojemnosci tworza tzw. *przestrzen adresowa mikroprocesora*. Zawartosc licznika rozkazów jest odkladana na tzw. stos w chwili, gdy pod wplywem akceptowanego przerwania zostaje zawieszona realizacja programu glównego lub w przypadku zastosowania specjalnego rozkazu przesylu zawartosci rejestru (np. *PUSH*) na stos. Zawartosc ta powraca do licznika rozkazów po zakonczeniu przerwania. *Stos* jest zespolem rejestrów sluzacym do doraznego magazynowania informacji. Charakteryzuje sie mozliwoscia szybkiego dostepu i organizacja typu LIFO (ang. *Last In First Out*), tzn. informacja odlozona jako ostatnia jest pobierana jako pierwsza. Stos umieszczony wewnatrz ukladu scalonego mikroprocesora ma zwykle pojemnosc do kilkunastu bajtów.

Znacznie wygodniejsze jest umieszczenie stosu w przestrzeni pamieci zapisywalnej (*RAM*). Wówczas jego miejsce w strukturze mikroprocesora zajmuje rejestr zwany *wskaznikiem stosu*. Rozkazy modyfikujace zawartosc wskaznika stosu umozliwiaja przemieszczanie stosu w przestrzeni adresowej pamieci zapisywalnej, tworzenie

w razie potrzeby kilku stosów itp. Pojemnosc stosu umieszczonego w przestrzeni *RAM* zalezy od decyzji programisty.

Informacje o sposobie wykonywania operacji sa przechowywane w rejestrze stanu. *Rejestr stanu* jest zespolem przerzutników przechowujacych tzw. *wskazniki stanu* (*znaczniki stanu*). Po zakonczeniu kolejnej operacji w ALU wskazniki zawieraja informacje o jej wyniku i sposobie wykonania. Zwykle sygnalizuje sie w ten sposób:

- zerowy lub niezerowy stan akumulatora,
- znak liczby otrzymanej w wyniku operacji,
- wystapienie przeniesienia lub pozyczki z najbardziej znaczacego bitu MSB (ang. *Most Significant Bit*),
- parzystosc lub nieparzystosc bitów jedynkowych w wyniku.

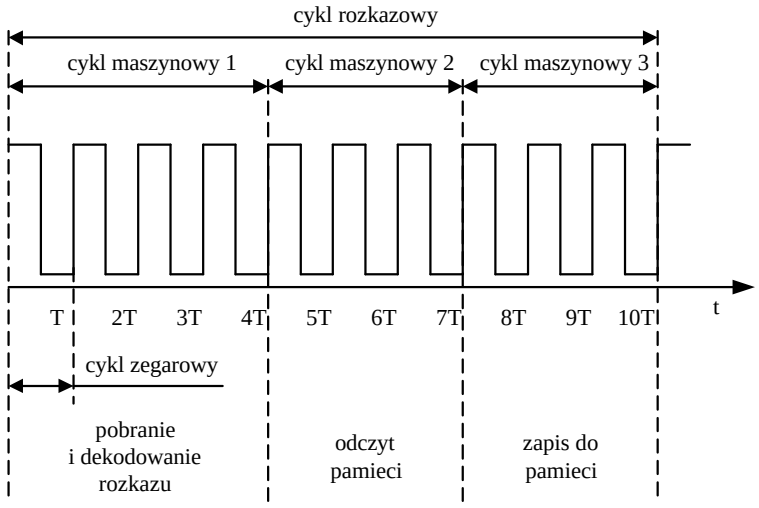
Stany wskazników moga wplywac na sposób wykonania nastepnej operacji w ALU lub na sposób wykonania kolejnych instrukcji programu. Dzieki temu jest mozliwa realizacja rozgalezienia programu za pomoca skoków warunkowych lub warunkowych odwolan do podprocedur.

Rejestr danych i rejestr adresów sluza do komunikacji mikroprocesora z innymi podzespolami systemu za posrednictwem magistrali danych i magistrali adresowej. Rejestry te sa wyposazone w stopnie mocy zapewniajace odpowiednia obciazalnoscia wyjsc mikroprocesora (zwykle ok. kilku mA). Transmisja slów adresowych jest jednokierunkowa, a transmisja danych przez rejestr danych moze sie odbywac zarówno od mikroprocesora do systemu, jak i w kierunku przeciwnym.

Praca mikroprocesora steruje zegar taktujacy. Najmniejszy przedzial czasu, jaki jest stosowany w ukladzie sterujacym, nosi nazwe *cyklu zegarowego*. Czas potrzebny na przeslanie slowa binarnego (slowa danych) miedzy mikroprocesorem i pamiecia lub urzadzeniem wejscia–wyjscia nazywa sie *cyklem maszynowym* (rys.1.6).

Cykl maszynowy obejmuje kilka cykli zegarowych i moze miec rózne dlugosci zaleznie od rodzajów wykonywanych operacji. Czas potrzebny na pobranie i realizacje jednego rozkazu nazywa sie *cyklem rozkazowym*. Obejmuje on od jednego do kilku cykli maszynowych, zaleznie od rodzaju rozkazu. Ze wzgledu na niezbedna szybkosc dzialania, w mikroprocesorach stosowanych w przemysle, dlugosc cyklu zegarowego

wynosi zazwyczaj kilkaset nanosekund, a dlugosc cyklu rozkazowego – od jednej do kilku mikrosekund



Rys.1.6. Cykl rozkazowy mikroprocesora

1.4. Elementy arytmetyki mikroprocesorów

Naturalnym dla czlowieka sposobem liczenia jest liczenie w systemie dziesietnym, tzn. kazda liczba jest zapisywana jako ciag cyfr o postaci:

$$a_n \ldots a_i \ldots a_1 \; a_0 \;,$$

gdzie dla kazdego i symbol a_i oznacza jedna z cyfr: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Wartosc tak zapisanej liczby mozna wyznaczyc ze wzoru:

$$W = \sum_{i=0}^{n} 10^i \, a_i = a_0 + 10 \, a_1 + \ldots + 10^n \, a_n \,.$$

Na kazdej pozycji liczby moze wystapic jedna z 10 cyfr, a stad wynika, ze przyrzad zdolny do zapamietania wartosci dowolnej cyfry w liczbie musialby miec az 10 róznych stanów. Elementarne komórki pamieci komputera (przerzutniki) moga przyjmowac tylko dwa stany, oznaczone umownie 0 i 1. Mozna jednak udowodnic, ze dwie cyfry sa wystarczajace do przedstawienia wszystkich liczb mozliwych do przedstawienia za pomoca dziesieciu cyfr dziesietnych. Liczba jest zapisywana jako ciag cyfr o postaci:

$$b_n \ldots b_i \ldots b_1 \quad b_0 \ ,$$

gdzie dla kazdego i symbol b_i oznacza jedna z cyfr 0, 1. Wartosc tak zapisanej liczby mozna wyznaczyc jako:

$$W = \sum_{i=0}^{n} 2^i b_i = b_0 + 2\ b_1 + \ldots + 2^n b_n.$$

Podany zapis liczb za pomoca dwóch cyfr nosi nazwe *zapisu dwójkowego* lub *binarnego.* Pojedyncza cyfre dwójkowa nazywa sie *bitem* (ang. *bit* – skrót od *binary digit*). Ciag bitów nazywa sie *slowem*. Bit b_0 jest bitem najmniej znaczacym oznaczanym LSB (ang. *Least Significant Bit*), bit b_n natomiast jest bitem najbardziej znaczacym MSB (ang. *Most Significant Bit*). Wartosc liczby dwójkowej, wyrazona w kodzie dziesietnym, uzyskuje sie przez sumowanie wartosci cyfr na poszczególnych pozycjach pomnozonych przez ich wagi.

Cala informacja przeznaczona dla mikroprocesora musi byc przedstawiona w postaci dwójkowej. Dotyczy to zarówno danych, stanowiacych argumenty rozkazów, jak i samych rozkazów, które musza byc zakodowane w postaci odpowiednich slów dwójkowych. Równiez wszystkie sygnaly elektryczne, za pomoca których mikroprocesor komunikuje sie z innymi ukladami wchodzacymi w sklad mikrokomputera, sa sygnalami dwustanowymi (dwuwartosciowymi). Jedna z wad zapisu binarnego jest znaczna dlugosc zapisanych liczb. Dlatego w technice mikroprocesorowej uzywa sie tzw. *zapisu szesnastkowego* (ang. *hexadecimal*), który mozna traktowac jako skrócona forme zapisu binarnego. W zapisie szesnastkowym za pomoca jednego znaku oznacza sie kolejne bity (tabela 1.1). Dla odróznienia od zapisu dziesietnego liczby w zapisie szesnastkowym oznacza sie litera H, np. 3AEH.

Poniewaz $16 = 2^4$, wiec przy przechodzeniu z zapisu szesnastkowego na binarny koduje sie oddzielnie kazda cyfre przy za pomoca 4 bitów, np.:

$$7AC3H = 0111\ \ 1010\ \ 1100\ \ 0011$$

$$96H = 1001\ \ 0110\ .$$

Dodatkowa zaleta zapisu szesnastkowego jest równy podzial 8–bitowego bajtu na dwie cyfry szesnastkowe.

Tabela 1.1. Zapis dziesietny, binarny i szesnastkowy cyfr 0–15

Dziesietny	0	1	2	3	4	5	6	7
Binarny	0000	0001	0010	0011	0100	0101	0110	0111
Szesnastkowy	0H	1H	2H	3H	4H	5H	6H	7H

Dziesietny	8	9	10	11	12	13	14	15
Binarny	1000	1001	1010	1011	1100	1101	1110	1111
Szesnastkowy	8H	9H	AH	BH	CH	DH	EH	FH

W technice mikroprocesorowej uzywany jest równiez zapis w *kodzie binarno-dziesietnym BCD* (ang. *Binary Coded Decimal*), w którym porcjom czterech kolejnych bitów przyporzadkowuje sie cyfry dziesietne od 0 do 9. Zastosowanie kodów BCD dla liczb dziesietnych polega na zastepowaniu kazdej cyfry dziesietnej odpowiadajacej jej porcji 4- bitowej, np.: w kodzie typu 8421, dla liczby dziesietnej 4718 bedzie:

$$4718_{\,d} \equiv 0100 \quad 0111 \quad 0001 \quad 1000_{\,BCD}$$

Koniecznosc stosowania róznych kodów w systemach mikroprocesorowych wynika stad, ze:

– natura liczb moze byc rózna, tzn. liczby calkowite dodatnie, liczby calkowite o róznych znakach, liczby, w których czesc calkowita i ulamkowa maja zawsze niezmienna lub zmienna liczbe pozycji;
– liczby pochodza z róznych zródel informacji, np.: z przetwornika analogowo-cyfrowego, klawiatury itp.;
– liczby maja rózne przeznaczenie, np.: sa to liczby na których beda wykonywane operacje arytmetyczne lub beda tylko wyswietlane.

Wlasciwy sposób kodowania liczb moze w istotny sposób uproscic realizacje ich przetwarzania lub przesylania. Jednostka arytmetyczno-logiczna wykonuje operacje arytmetyczne na liczbach binarnych, które moga byc traktowane jako:

– liczba calkowita dodatnia zapisana w *naturalnym kodzie binarnym*,
– liczba calkowita ze znakiem zapisana w *kodzie uzupelnien do dwóch*,
– liczba calkowita bez znaku w *zapisie dwójkowo-dziesietnym BCD*.

Naturalny kod binarny odpowiada zapisowi liczby w systemie dwójkowym. W 8–bitowym rejestrze mozna umiescic liczby w zapisie kodu binarnego z przedzialu (0, 255), w 16–bitowym z przedzialu (0, 65535).

W kodzie uzupelnien do dwóch mozna zapisywac liczby calkowite ze znakiem. Liczbe *n*–bitowa w tym kodzie mozna zapisac nastepujaco:

$$a = -a_{n-1}2^{n-1} + \sum_{i=0}^{n-2} a_i 2^i .$$

Najbardziej znaczacy bit liczby ma wiec wage -2^{n-1} i okresla jej znak. Jesli $a_{n-1} = 0$, czyli liczba jest dodatnia, zapis w kodzie uzupelnienia nie rózni sie od zapisu w kodzie binarnym. W 8–bitowym rejestrze mozna umiescic liczbe w kodzie uzupelnienia z przedzialu ($-2^7 \div +2^7-1$) czyli (–128, +127). Zapis w tym kodzie ma kilka cech, które powoduja, ze jest on powszechnie stosowany w technice komputerowej, a mianowicie:

– dzialania (dodawanie i odejmowanie) na liczbach w zapisie uzupelnienia do dwóch sa analogiczne do liczb w kodzie binarnym;
– zmiana znaku liczby polega na zanegowaniu wszystkich bitów i dodaniu jedynki (np. 7 = 0111, –7 = 1001);
– liczby dodatnie maja identyczna postac jak w kodzie binarnym.

Podstawowa operacja na liczbach dwójkowych jest dodawanie dwóch slów. W przypadku dodawania do siebie dwóch bajtów wynik moze byc liczba wieksza od mozliwej do zakodowania w jednym bajcie, np.:

A =	01001101	77
B =	11000111	+199
A + B	00010100	276>255
C = 1		

Poniewaz wynik nie miesci sie w 8–bitowym slowie, wiec pojawia sie *bit przeniesienia* **C** (ang. *carry*) z najbardziej znaczacego bitu. Pojawienie sie przeniesienia **C** przy dodawaniu dwóch liczb w kodzie dwójkowym sygnalizuje przekroczenie zakresu przez wynik. Odejmowanie liczb dodatnich (bez znaku) jest realizowane przez dodanie do odjemnej uzupelnienia do dwóch odjemnika. Uzupelnienie do dwóch liczby binarnej

uzyskuje sie przez zanegowanie wszystkich bitów liczby i nastepnie dodanie jedynki. Jezeli zostanie pominiete przeniesienie, to wynik jest prawidlowy. Podczas odejmowania negacja bitu przeniesienia $\overline{C}=1$ sygnalizuje wystapienie *pozyczki* (ang. *borrow*) na najbardziej znaczacym bicie, czyli przekroczenie zakresu. Bit przeniesienia moze byc równiez wykorzystany przy porównywaniu dwóch liczb calkowitych bez znaku.

A =	01000110	70
B =	00110100	–52
A – B	000110010	18
C = 1		

Dzialania dodawania i odejmowania liczb calkowitych ze znakiem w zapisie uzupelnien do dwóch sa takie same jak powyzej, zmienia sie tylko warunek przekroczenia zakresu przez wynik. Przekroczenie zakresu przez liczby w zapisie uzupelnien do dwóch nazywa sie *nadmiarem* **OV** (ang. *overflow*). Nadmiar jest suma modulo 2 przeniesien C_n i C_{n-1}:

$$\mathbf{OV} = C_n \oplus C_{n-1} = \overline{C_n} C_{n-1} \vee C_n \overline{C_{n-1}} \ .$$

	01111000	120		11101101	–19
+	01101001	+105	+	10001111	–113
$C_n = 0$	11100001	225>127	$C_n = 1$	01111100	–132<–128
$C_{n-1} = 1$	OV = 1		$C_{n-1} = 0$	OV = 1	

Dodawanie liczb w kodzie BCD jest realizowane podobnie jak w naturalnym kodzie binarnym. Jednakze po kazdej operacji konieczne jest sprawdzenie, czy otrzymana czterobitowa liczba reprezentujaca liczbe dziesietna nie jest wieksza od 9. Jezeli tak, to konieczne jest wykonanie *korekcji dziesietnej* (ang. *decimal adjust*) wyniku, polegajacej na dodaniu do niego liczby binarnej 6 (0110). Wskaznikiem koniecznosci wykonania korekcji jest wystapienie przeniesienia z najbardziej znaczacego bitu cyfry dziesietnej lub przekroczenie przez liczbe reprezentujaca wynik wartosci 9, np.:

	0101	
+	0110	
	1011	>1001
+	0110	korekcja
1	0001	

	1000	
+	1001	
1	0001	przeniesienie
+	0110	korekcja
1	0111	

W przykladach wystapilo przeniesienie do bardziej znaczacej cyfry BCD. Podczas dodawania liczb wielocyfrowych jest konieczne wykonywanie korekcji dziesietnej kolejno dla wszystkich cyfr, poczawszy od najmniej znaczacej. W przypadku wystapienia przeniesienia, powinno byc ono dodane do cyfry o rzad bardziej znaczacej.

Cechy charakterystyczne wyników operacji wykonywanych w jednostce arytmetyczno-logicznej sa zapamietywane w rejestrze znaczników (rejestrze statusowym). Wybrane bity tego rejestru informuja o wyniku wykonywanych dzialan:

- Znacznik przekroczenia **C** sygnalizuje przekroczenie zakresu 8–bitowych liczb calkowitych bez znaku w operacji dodawania i odejmowania oraz przekroczenie zakresu 0...99 podczas operacji korekcji dziesietnej. Oznacza przeniesienie miedzy kolejnymi dodawanymi bajtami lub pozyczke przy wielobajtowym odejmowaniu.
- Znacznik przeniesienia polówkowego **AC** ma podobne znaczenie jak znacznik przeniesienia **C**, ale dotyczy przeniesienia miedzy czterema mniej znaczacymi i czterema bardziej znaczacymi bitami akumulatora. Znacznik jest stosowany tylko w operacjach wykonywanych dla liczb zapisanych w kodzie BCD, którym towarzyszy korekcja dziesietna.
- Znacznik nadmiaru **OV** informuje o przekroczeniu zakresu 8–bitowych liczb calkowitych ze znakiem, –128, +127 w operacjach dodawania i odejmowania.
- Stan znacznika parzystosci **P** jest ustawiany po kazdej instrukcji zmieniajacej stan akumulatora w taki sposób, aby liczba jedynek w akumulatorze i znaku parzystosci byla parzysta.

1.5. Tryby adresowania pamieci

Uklad sterowania mikroprocesora dziala cyklicznie wykonujac *cykle rozkazowe*. Wyróznia sie dwie fazy cyklu rozkazowego: faza *pobrania kodu* rozkazu oraz faza *wykonania rozkazu*. Faza pobrania przebiega tak samo dla wszystkich rozkazów. Kolejne rozkazy sa pobierane z kolejnych komórek pamieci programu i do adresowania tej pamieci w fazie pobrania sluzy licznik rozkazów PC, którego zawartosc jest zwiekszana o 1 po kolejnym pobraniu. Pobrany rozkaz jest przesylany do rejestru rozkazów,

a z niego do ukladu sterowania. Wykonanie rozkazu moze polegac na przetworzeniu informacji zawartej w rejestrach procesora albo w pamieci lub na przeslaniu informacji miedzy rejestrem procesora a pamiecia lub ukladem wejscia−wyjscia. Faza wykonania przebiega róznie dla róznych rozkazów. Dla kazdego rozkazu uklad sterowania generuje odpowiednie sekwencje sygnalów sterujacych, powodujacych jego wykonanie.

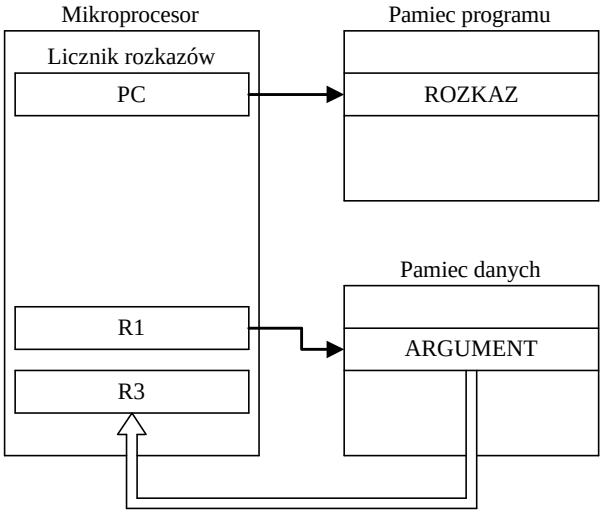
Zbiór rozkazów typowego mikroprocesora mozna podzielic na cztery zasadnicze grupy:

– rozkazy sluzace do przesylania danych miedzy rejestrami procesora lub miedzy rejestrami a pamiecia;
– rozkazy sluzace do wykonywania operacji matematycznych i logicznych na zawartosci rejestrów roboczych lub pamieci;
– rozkazy sterujace wykonywaniem programu;
– rozkazy wejscia–wyjscia.

Przesylanie informacji miedzy rejestrami mikroprocesora oparte jest o tzw. rozkaz bezadresowy. Jego wykonanie nie wymaga odwolywania sie do modulów zewnetrznych. Przesylanie informacji natomiast miedzy rejestrem a pamiecia wymaga tzw. rozkazu adresowego i dlatego konieczne jest okreslanie adresu efektywnego komórki pamieci. Wiaze sie to z tzw. *trybami adresowania pamieci* (ang. *addressing mo*de). Tryb adresowania okresla miejsce, gdzie umieszczany jest adres argumentu lub sposób w jaki jest on obliczany.

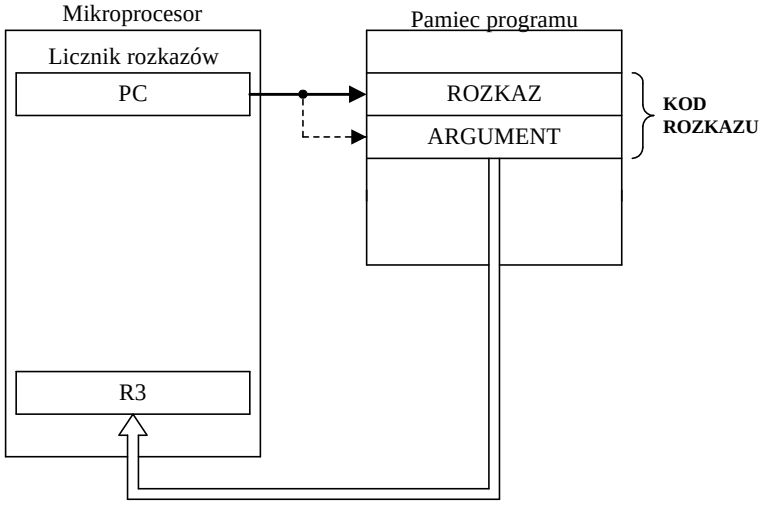
Adresowanie posrednie (ang. *indirect addressing*) odnosi sie do rozkazów zawierajacych poza kodem operacji adres komórki pamieci, w której znajduje sie adres argumentu. Odmiana adresowania posredniego jest tzw. adresowanie za pomoca *wskazników* (ang. *pointer addressing*) zwane inaczej *adresowaniem rejestrowym posrednim* lub *adresowaniem zawartoscia rejestrów*. Odnosi sie ono do rozkazów, które swoim kodem wskazuja rejestr lub rejestry zawierajace adres argumentu rozkazu. Rejestry te nazywa sie *rejestrami wskaznikowymi* lub *licznikiem danych*. Na zawartosci rejestru wskazników mozna wykonywac operacje arytmetyczne (np. dodawanie jednosci) czyli modyfikowac adres. Mozliwosc modyfikacji adresu jest wazna cecha trybu adresowania, ulatwia bowiem wykonywanie operacji na zlozonych strukturach da-

nych. Na rysunku 1.7 przedstawiono schematycznie adresowanie za pomoca wskaznika. Przykladowo w fazie wykonania, na magistrale adresowa jest wysylana zawartosc rejestru **R1**, a slowo z pamieci (zwane argumentem, operandem) jest podawane na magistrale danych i wpisywane do **R3**.



Rys. 1.7. Adresowanie za pomoca wskaznika

Innym sposobem adresowania pamieci jest *adresowanie natychmiastowe* (ang. *immediate addressing*) przedstawione na rys. 1.8.

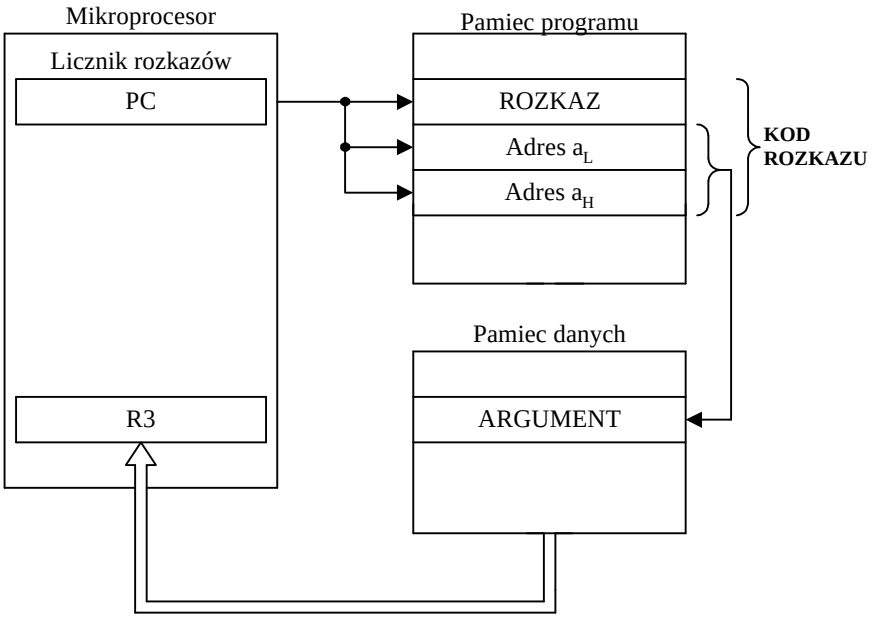


Rys. 1.8. Adresowanie natychmiastowe (z argumentem bezposrednim)

W tym przypadku argument (np. stala) jest umieszczony w pamieci programu bezposrednio za kodem rozkazu, czyli po pobraniu kodu rozkazu adres argumentu jest

zawarty w liczniku rozkazów. Rozkazy o adresowaniu natychmiastowym nazywa sie rozkazami z argumentem bezposrednim. W fazie wykonywania rozkazu zawartosc licznika rozkazów **PC** jest automatycznie zwiekszana o 1 (podobnie jak po pobraniu kodu operacji), tak aby po wykonaniu licznik rozkazów zawieral adres nastepnego rozkazu.

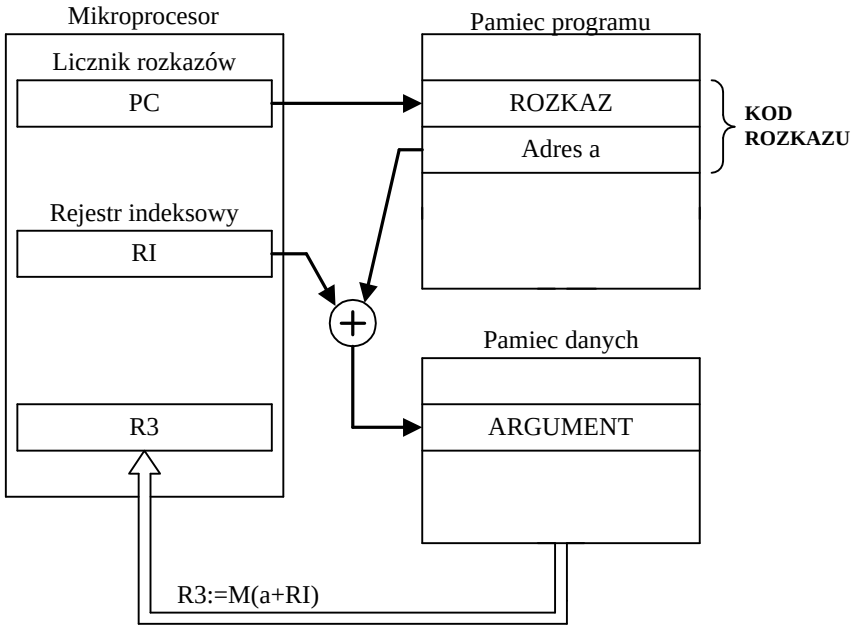
Na rysunku 1.9 przedstawiono schemat *adresowania bezposredniego* (ang. *direct addressing*). Przy takim adresowaniu adres argumentu jest umieszczany w pamieci programu, w slowie nastepujacym za rozkazem. Szesnastobitowy adres (mikroprocesorów 8 bitowych) zajmuje dwa kolejne slowa pamieci. W pierwszym jest umieszczana mniej znaczaca czesc adresu a_L (bity 0–7), a w drugim – bardziej znaczaca a_H (bity 8–15). Adresowanie bezposrednie jest czesto stosowane w rozkazach skoku oraz do adresowania danych zajmujacych pojedyncze slowa pamieci.



Rys. 1.9. Adresowanie bezposrednie

Przy *adresowaniu indeksowym* (ang. *index addressing*) adres argumentu otrzymuje sie przez dodanie adresu bezposredniego, umieszczonego za rozkazem, do zawartosci rejestru procesora np. **RI** (wskaznika danych), zwanego w tym przypadku *rejestrem indeksowym*. Schemat adresowania indeksowego przedstawiono na rys. 1.10.

Jak widac, adresowanie indeksowe jest polaczeniem adresowania zawartoscia rejestru z adresowaniem bezposrednim. Istnieje tu mozliwosc latwej modyfikacji adresu.



Rys. 1.10. Adresowanie indeksowe

Stosowane jest przede wszystkim do adresowania tablic umieszczonych w pamieci. Jezeli adres poczatku tablicy jest staly, a polozenie elementu wzgledem tego poczatku (przesuniecie) trzeba obliczac, to nalezy ten adres umiescic za rozkazem, a przesuniecie w rejestrze indeksowym. W wielu przypadkach polozenie elementu struktury wzgledem jej poczatku jest stale (np. odleglosc elementu od wierzcholka stosu), natomiast adres poczatku jest obliczany w trakcie wykonywania programu. Wówczas nalezy przesuniecie umiescic za rozkazem, a adres poczatku struktury – w rejestrze indeksowym. Czesto tak uzyty rejestr indeksowy jest nazywany *rejestrem bazowym*, a tryb adresowania – *adresowaniem bazowym*.

1.6. Uklady otoczenia mikroprocesora

1.6.1. Wprowadzenie

Przez pojecie *uklady otoczenia mikroprocesora* okresla sie zwykle te czesci systemu mikroprocesorowego, które w sposób bezposredni wspólpracuja z jednostka centralna *CPU*. Sa to wiec: uklady *pamieci programu i danych* oraz *urzadzenia wejsciowe* i *wyjsciowe***.** Przemyslowe sterowniki mikroprocesorowe, czyli tzw. mikrokontrolery, stosowane do sterowania bezposredniego, naleza do grupy malych systemów. Przy ich projektowaniu znaczna uwage przywiazuje sie do uproszczenia struktury i obnizki ceny jednostkowej. Dazy sie równiez do maksymalnego uwzglednienia wymagan stawianych przez konkretny obiekt regulacji. W wyniku tych uproszczen powstaja struktury rózniace sie od rozwiazan spotykanych w mikrokomputerach ogólnego przeznaczenia.

Pomimo róznorodnosci zastosowan daje sie zauwazyc dazenie do unifikacji systemów i podzespolów mikroprocesorowych sterowników przemyslowych. Próbuje sie ujednolicic jednostki centralne, rozmiary pamieci, liczbe i rodzaj wejsc pomiarowych i wyjsc sterujacych. Dazenie do pewnej unifikacji mikrokontrolerów i ich podzespolów wynika równiez z potrzeby obnizenia ceny i ulatwienia czynnosci serwisowych.

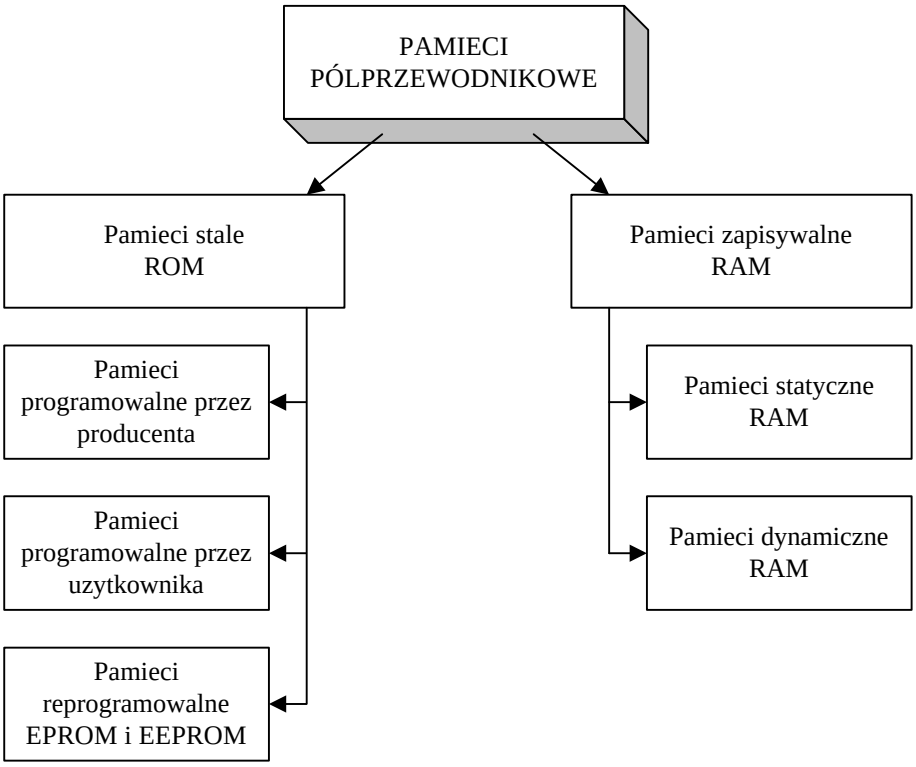
1.6.2. Uklady pamieci

Pamiec systemu mikroprocesorowego mozna podzielic ze wzgledu na wlasciwosci i przeznaczenie na dwie odrebne czesci: ***pamiec zapisywalna*** i ***pamiec stala***. Kazda z tych czesci sklada sie z pamieciowych ukladów scalonych, polaczonych w sposób zalezny od ich organizacji wewnetrznej. Na rysunku 1.11 pokazano podzial funkcjonalny pamieci pólprzewodnikowych.

Pamiec zapisywalna RAM **(***ang.* ***R****andom* ***A****ccess* ***M****emory*) sluzy do przechowywania danych pomiarowych, programów, które moga byc zmieniane oraz przejsciowych wyników obliczen. Mikroprocesor moze zatem z pamieci *RAM* czytac oraz do niej zapisywac informacje. Jest to zwykle pamiec ulotna. Zapisana w niej informacja zanika przy braku napiec zasilajacych. Po kazdym zalaczeniu zasilania informacja ta musi byc odtwarzana.

Ta cecha powoduje, ze niemozliwe jest stosowanie w systemach mikroproceso-

rowych tylko takich pamieci, poniewaz kazdy zanik napiecia zasilajacego powodowalby wymazywanie programu i koniecznosc ponownego jego wprowadzania.



Rys. 1.11. Podzial funkcjonalny pamieci pólprzewodnikowych

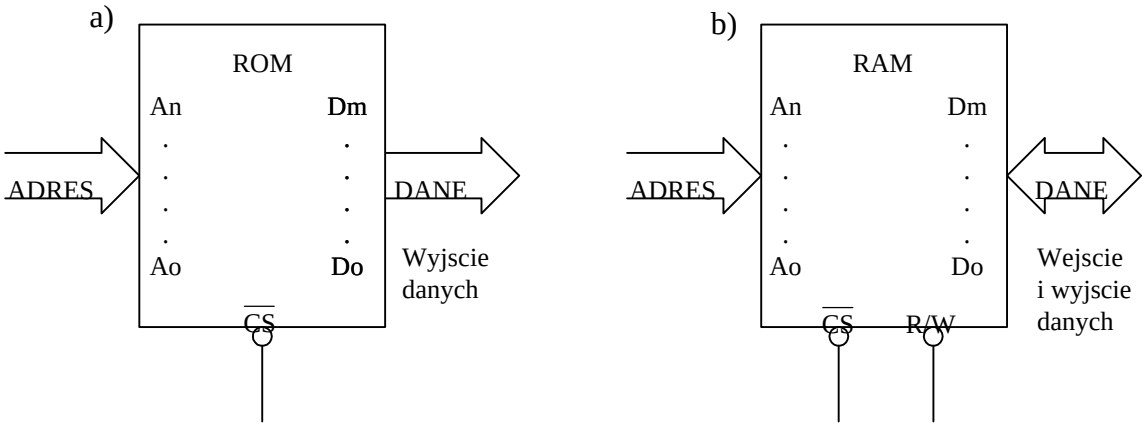
Dlatego w systemach mikroprocesorowych zawsze obok pamieci *RAM* stosuje sie *pamiec stala ROM* (*ang.* ***R**ead **O**nly **M**emory*), która nie traci informacji po wylaczeniu napiecia zasilania, ale zapis do niej informacji wymaga specjalnych warunków i nie jest mozliwy w czasie normalnej pracy mikroprocesora. W mikrokomputerach specjalizowanych, które przez caly czas wykonuja taki sam program, jest on umieszczony wlasnie w pamieci typu *ROM*, dzieki czemu niemozliwe jest jego przypadkowe zniszczenie. W mikrokomputerach uniwersalnych, w których wykonywany program uzytkowy czesto jest zmieniany, do pamieci typu *ROM* jest wpisywany program zarzadzajacy typu monitor systemu, umozliwiajacy tworzenie lub modyfikacje programu uzytkowego. Zapisywanie w pamieci *ROM* moze odbywac sie u wytwórcy w czasie procesu technologicznego.

Odmiana *PROM* (ang. ***P**rogrammable ROM*) moze byc jednorazowo zaprogramowana u uzytkownika. Tu równiez niemozliwa jest zmiana raz wpisanego do niej

programu. Odmiana *EPROM* (ang. ***E**rasable PROM*) moze byc wielokrotnie programowana u uzytkownika za pomoca specjalnego programatora. Zawartosc pamieci mozna skasowac przez naswietlanie promieniami ultrafioletowymi, po czym mozliwe jest ponowne jej zaprogramowanie. Sa to wiec pamieci reprogramowalne.

Istnieja równiez pamieci *EEPROM* (ang. ***E**lectrically EPROM*), których kasowanie zawartosci odbywa sie elektrycznie. Sposób programowania pamieci *PROM*, *EPROM* i *EEPROM* zalezy od jej typu i jest zawsze podawany przez producenta. Zwykle programowanie polega na podaniu na jedno z wejsc napiecia programujacego (w zaleznosci od typu 12V-13V lub w starszych rozwiazaniach 25V−30 V), a na inne kilkudziesieciomilisekundowych impulsów programujacych.

Pamieci typu *RAM* dziela sie na ***statyczne*** i ***dynamiczne***. Podzial ten wynika z technologii wykonania wplywajacej jednak istotnie na sposób wykorzystania pamieci. Pamieci dynamiczne maja zwykle wieksza pojemnosc (jednego modulu scalonego) niz pamieci statyczne. Aby nie tracily swojej zawartosci, wymagaja one odswiezania, polegajacego na podaniu impulsu na specjalne wejscie.



Rys. 1.12. Schematy funkcjonalne typowych modulów pamieci:
a) typu ROM, b) typu RAM

Na rysunku 1.12. przedstawiono schematy funkcjonalne typowych modulów pamieci pólprzewodnikowej typu *ROM* i *RAM*. Kazdy modul pamieci typu *ROM* (rys. 1.12. a) ma pewna liczbe *wejsc adresowych* **A**, przez które jest podawany adres komórki pamieci oraz pewna liczbe *wyjsc danych* **D**, na które jest podawana zawartosc

komórki pamieci o adresie podanym na wejscie **A**. Oprócz tego kazdy modul ma dodatkowe *wejscie wybierajace* (ang. ***C**hip **S**elect* lub ***C**hip **E**nable*) – $\overline{\mathbf{CS}}$ ($\overline{\mathbf{CE}}$) – jedno lub wiecej. Modul pamieci pracuje (wysyla dane na wyjscie) tylko wtedy, gdy na wejscie $\overline{\mathbf{CS}}$ jest podany sygnal o niskim poziomie ($\overline{\mathbf{CS}}$ **= 0**). Pamieci typu *RAM* (rys. 1.12. b) maja ponadto *wejscie rodzaju pracy* **R** / $\overline{\mathbf{W}}$ ($\overline{\mathbf{WE}}$) (*ang. **R**ead/**W**rite* lub ***W**rite **E**nable*), przy czym zwykle **R** / $\overline{\mathbf{W}}$ **= 1** okresla operacje odczytu z pamieci, $\overline{\mathbf{R} / \mathbf{W}}$ **= 0** okresla operacje zapisu do pamieci. Operacja zapisu lub odczytu bedzie wykonana tylko wtedy, gdy $\overline{\mathbf{CS}}$ **= 0**. Wejscie danych do pamieci typu *RAM* jest zwykle wspólne z wyjsciem danych. Odpowiednie wyprowadzenia zachowuja sie jak wyjscia lub jak wejscia w zaleznosci od rodzaju pracy (stan **R** / $\overline{\mathbf{W}}$ przy $\overline{\mathbf{CS}}$ **= 0**) lub sa w stanie duzej impedancji, gdy $\overline{\mathbf{CS}}$ **= 1**. Niektóre moduly pamieci maja kilka wejsc uaktywniajacych $\overline{\mathbf{CS}}$ lub wystepuje dodatkowe wejscie $\overline{\mathbf{OE}}$ (ang. ***O**utput **E**nable*), umozliwiajace odlaczenie wyjscia danych **D** od magistrali systemu (otwiera lub zamyka wyjsciowy bufor danych w celu synchronizacji czynnosci odczytu).

Podstawowymi parametrami charakteryzujacymi modul pamieci sa: pojemnosc, organizacja, czas dostepu.

Pojemnoscia modulu pamieci nazywa sie liczbe bitów informacji, jakie mozna do niej zapisac (np. 1 kbit = 2^{10} = 1024 bity).

Organizacja pamieci jest to sposób rozmieszczenia bitów w slowa. Na przyklad modul pamieci o pojemnosci 8 kbitów moze byc zorganizowany miedzy innymi jako:

– 1024 x 8 bitów, co oznacza 1024 slowa osmiobitowe,
– 2048 x 4 bity, co oznacza 2048 slów czterobitowych.

Organizacja modulów pamieci decyduje o sposobie dolaczenia ich do systemu mikroprocesorowego. Przykladowo, budujac z modulów 2048 x 4 pamiec dla mikrokomputera 8–bitowego, nalezy zastosowac dwa moduly, aby uzyskac 2048 slów 8–bitowych.

Czas dostepu t_D jest parametrem charakteryzujacym szybkosc dzialania pamieci. Jest to czas, jaki uplywa od chwili podania adresu na wejscie **A** do chwili pojawienia sie na wyjsciu danych z komórki pamieci o podanym adresie (przy $\overline{\mathbf{CS}}$ **= 0**). Czas do-

stepu jest zawsze podawany przez producenta pamieci jako podstawowy parametr. Moduly pamieci uzyte do budowy mikrokomputera musza miec czas dostepu mniejszy niz wymagany przez uzyty mikroprocesor.

Obecnie produkuje sie bardzo duzo róznych typów modulów pamieci, z których najpopularniejsze to: pamieci RAM 2114 (1024 slowa x 4 bity) lub najnowsze 6264 i 6116 (2048 slów x 8 bitów), firmy INTEL oraz pamieci EPROM 2716 (2048 slów x 8 bitów) lub 27C256 i 2764 (8192 slów x 8 bitów), równiez firmy INTEL.

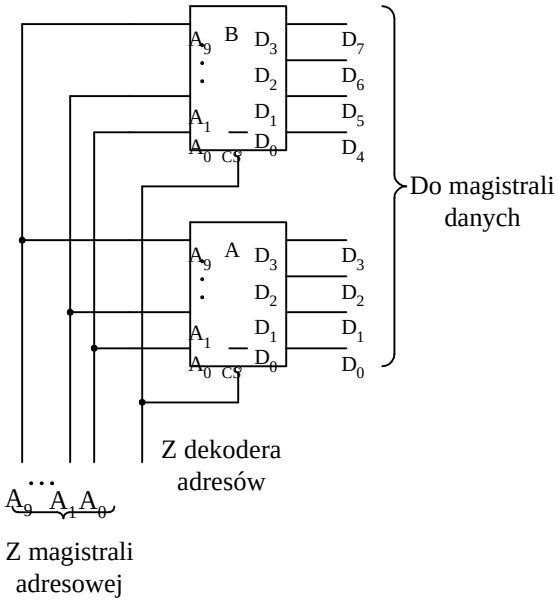
W zaleznosci od potrzeb, z modulów scalonych pamieci jest budowana pamiec systemu mikroprocesorowego. Sposób budowy pamieci zalezy od:

- typu posiadanych modulów, ich pojemnosci i organizacji;
- wymaganej mapy pamieci, czyli podzialu przestrzeni adresowej mikrokomputera na czesci przeznaczone na pamiec typu *ROM* i *RAM* i czesc niewykorzystana.

Pamiec systemu mikroprocesorowego powinna spelniac nastepujace warunki:

– dlugosc slowa pamieci powinna byc równa dlugosci slowa magistrali danych,

– komórki pamieci powinny byc jednoznacznie adresowane tzn. jednemu adresowi odpowiada zawsze jedna i ta sama komórka pamieci.

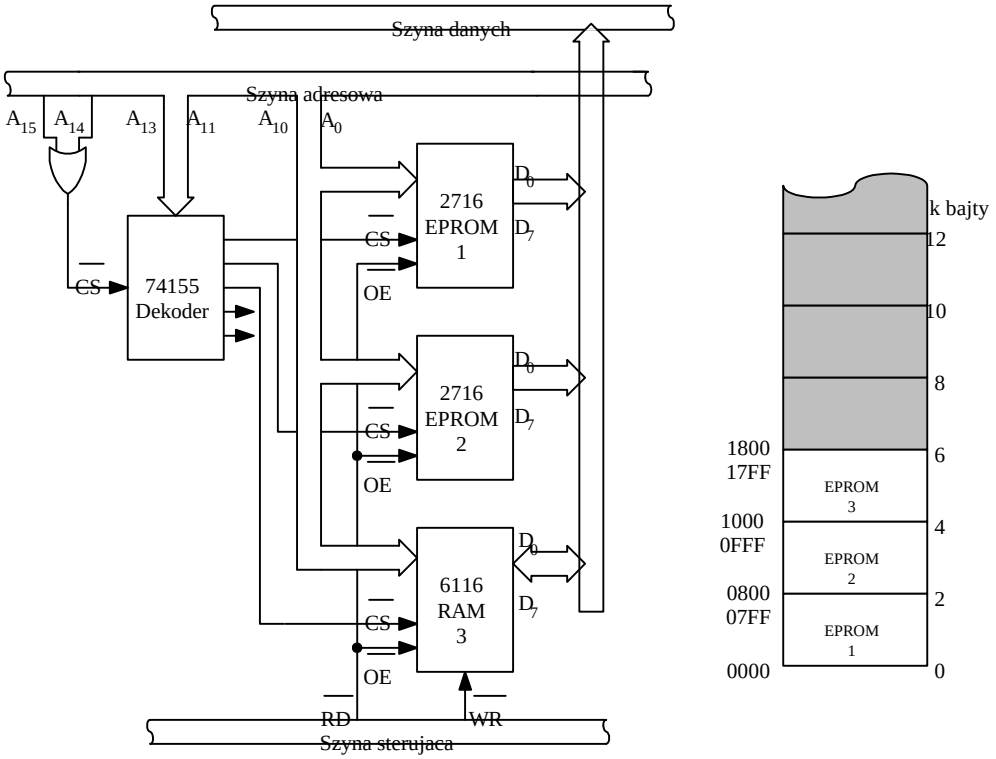


Rys. 1.13. Przyklad równoleglego laczenia modulów pamieci

Dla zapewnienia odpowiedniej dlugosci slowa, jezeli moduly pamieci maja slowa krótsze, laczy sie je równolegle. Na rysunku 1.13 przedstawiono równolegle polaczenie dwóch modulów pamieci o organizacji 1024x4 konieczne do uzyskania

8–bitowego slowa. Wejscia **A** i $\overline{\mathbf{CS}}$ modulów sa polaczone równolegle, natomiast wyjscia modulu A tworza bity 0–3, a wyjscia modulu B bity 4–7 slowa pamieci i beda podlaczone do szyny danych mikrokomputera. W identyczny sposób mozna tworzyc slowa pamieci o dowolnej dlugosci.

Aby uzyskac jednoznacznosc adresowania pamieci dzieli sie magistrale adresowa na dwie czesci: mniej znaczace bity adresu wybieraja komórke pamieci wewnatrz modulu i sa dolaczane do wejsc **A**, bardziej znaczace bity adresu wybieraja modul. Wybieranie modulu pamieci na podstawie bardziej znaczacych bitów adresu jest nazywane *dekodowaniem adresów*. Dla przykladu na rys. 1.14 przedstawiono budowe bloku pamieci skladajacego sie z modulów EPROM 2716 i RAM 6116, kazdy o pojemnosci 2048 bajtów.

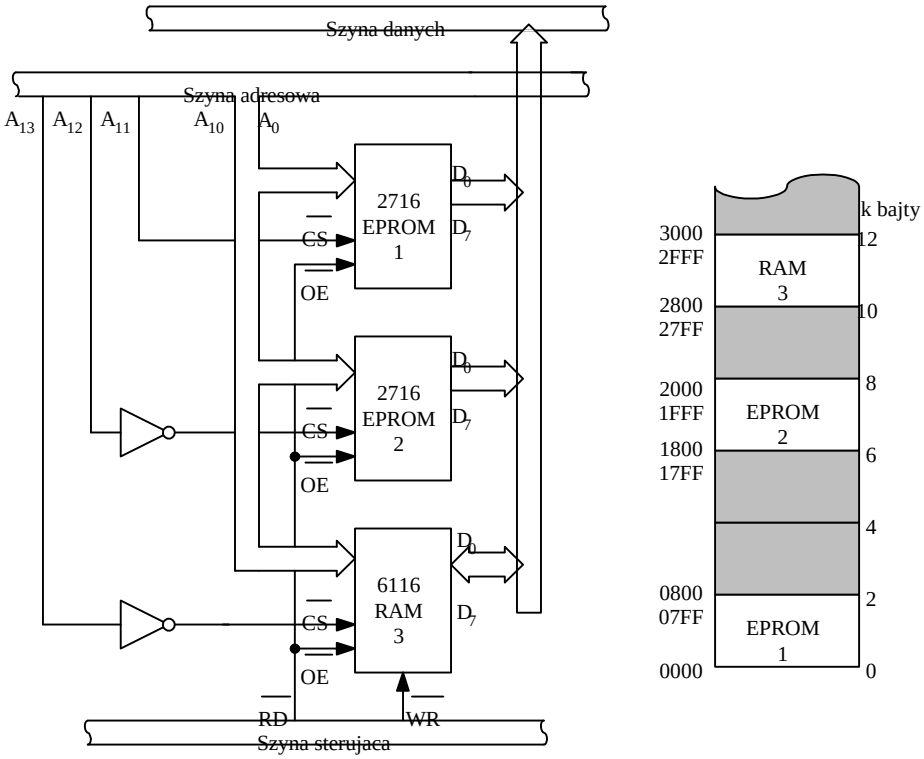


Rys. 1.14. Przyklad laczenia modulów pamieci i mapa pamieci przy pelnym dekodowaniu adresów

Obok schematu polaczen umieszczono mape pamieci, tzn. przyporzadkowanie przestrzeni adresowej poszczególnym ukladom scalonym. Bity A_0–A_{10} magistrali adresowej sa dolaczone do wejsc adresowych kazdego z modulów i wybieraja komórke

wewnatrz modulu. Bardziej znaczace bity magistrali adresowej A_{11}–A_{15} sa podawane na dekoder, którego wyjscia sa polaczone z wejsciami $\overline{\mathbf{CS}}$ kolejnych modulów. W ten sposób w danej chwili bedzie pracowal tylko ten modul, którego numer jest podany na liniach A_{11}–A_{15} magistrali adresowej. W przykladzie zastosowano tzw. *pelne dekodowanie adresów*, konieczne wtedy, gdy jest wykorzystywana cala przestrzen adresowa (na mapie pamieci kolejnym bajtom informacji przechowywanej w bloku sa przyporzadkowane kolejne adresy od 0000H do 17FFH wlacznie).

Gdy pojemnosc pamieci niezbednej w systemie mikroprocesorowym jest mniejsza niz przestrzen adresowa, mozna zastosowac *dekodowanie czesciowe*.

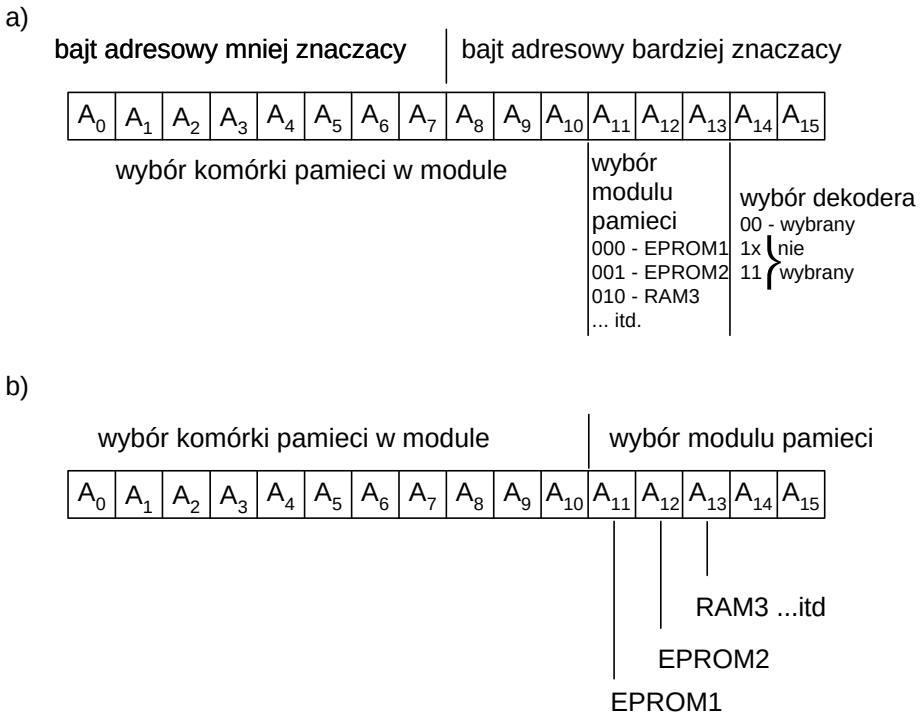


Rys. 1.15. Przyklad laczenia modulów pamieci i mapa pamieci przy czesciowym dekodowaniu adresów (selekcja liniowa)

W ukladzie jak na rys. 1.15 zastosowano tzw. *selekcje linowa.* Poszczególnym ukladom scalonym przyporzadkowano kolejne starsze bity adresowe A_{11}, A_{12}, A_{13}, eliminujac potrzebe zastosowania dekodera. W tym przypadku mapa pamieci traci cia-

gly charakter. W bloku wykorzystuje sie adresy od 0000H do 07FFH, od 1800H do 1FFFH oraz od 2800H do 2FFFH wlacznie. Pozostale czesci pamieci zostaja utracone. Sposób selekcji liniowej stosuje sie w malych, przemyslowych systemach mikroprocesorowych w celu obnizenia kosztów urzadzen.

W zaleznosci od przyjetego sposobu selekcji modulów pamieci rózny jest format slowa adresowego. Selekcji z pelnym dekodowaniem odpowiada format slowa przedstawiony na rys. 1.16a.



Rys. 1.16. Formaty slów adresowych pamieci dla przykladów z rys. 1.14 i 1.15.
a) selekcja z pelnym dekodowaniem, b) selekcja liniowa

Bity od A_0 do A_{10} sluza do wyboru adresu wewnatrz modulów. Ta czesc slowa jest dekodowana przez wewnetrzne selektory wierszy i kolumn w kazdym module pamieci. Bity A_{11}, A_{12}, A_{13} sa dekodowane przez scalony dekoder 74155. Mozliwe jest wiec odróznienie osmiu modulów, z których kazdy ma pojemnosc 2–kilobajtów. Najbardziej znaczace bity A_{14} i A_{15} mozna wykorzystac do wyboru jednego z czterech równorzednych dekoderów (za pomoca dodatkowego dekodera nadrzednego). Mozna

wiec docelowo wybierac jeden z 32 modulów pamieci o pojemnosci 2 kbajtów, co wypelnia cala mozliwa przestrzen mikroprocesora 8–bitowego (tzn. 64 kilobajty). W przypadku selekcji liniowej ulega zmianie sposób dekodowania pieciu najbardziej znaczacych bitów slowa adresowego (rys. 1.16b). Kazdemu modulowi pamieci jest przyporzadkowany jeden z bitów A_{11}–A_{15}. Wobec tego istnieje mozliwosc wyboru jednego z pieciu modulów pamieci o pojemnosci 2 kilobajtów, co oznacza mozliwosc wykorzystania jedynie czesci przestrzeni adresowej o pojemnosci 10 kilobajtów. W kazdym przypadku o wybraniu lub niewybraniu danego modulu decyduje stan tego bitu, który zostal mu przyporzadkowany. Pozostale cztery z pieciu najbardziej znaczacych bitów powinny odpowiadac stanowi niewybrania.

1.6.3. Uklady wejscia - wyjscia

Uklady wejscia–wyjscia sluza do transmisji danych miedzy mikroprocesorem a urzadzeniami zewnetrznymi. Ze wzgledu na ich posredniczenie w wymianie informacji, czesto sa nazywane ukladami sprzegajacymi mikroprocesor z urzadzeniami zewnetrznymi. W ukladach automatyki zwykle uklady wejscia–wyjscia posrednicza w wymianie informacji miedzy systemem mikroprocesorowym a obiektem regulacji. Zwykle sygnalami wejsciowymi sa wielkosci zadane i wielkosci mierzone, sygnalami zas wyjsciowymi wielkosci nastawiajace i pomocnicze sygnaly sterujace. Najprostszym sprzegajacym uniwersalnym ukladem wyjsciowym moze byc rejestr, do którego mikroprocesor wpisuje informacje przeznaczona do wyslania na zewnatrz. Natomiast zespól bramek trójstanowych moze byc prostym ukladem wejsciowym. Uklady te moga byc stosowane do laczenia mikroprocesora z urzadzeniami dwustanowymi, takimi jak: wylaczniki krancowe, przekazniki, lampki lub diody sygnalizacyjne itp. Sygnaly TTL pojawiajace sie na wyjsciu rejestru powinny byc wzmocnione przed doprowadzeniem do cewki przekaznika lub lampki. Moga równiez byc zamienione na postac analogowa za pomoca przetwornika cyfrowo-analogowego. Tego typu wejscia–wyjscia nosza nazwe *wejsc-wyjsc bezposrednich.*

W ogólnym przypadku mozna wyróznic dwa podstawowe rodzaje ukladów wej-

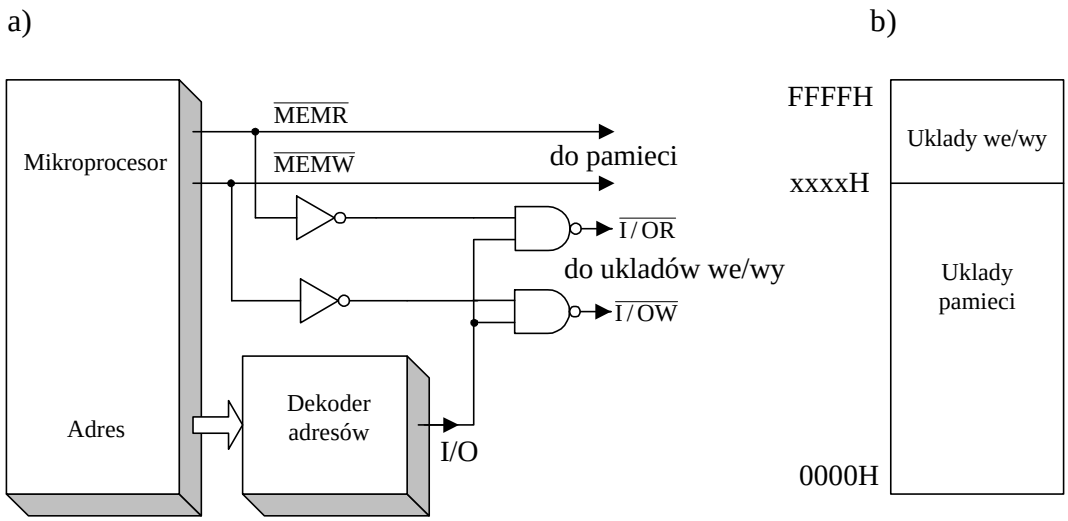
scia–wyjscia:

– uniwersalne uklady wejscia-wyjscia, przeznaczone do realizacji zadan sterowania poprzez wpis do ich odpowiednich rejestrów sterujacych danych okreslajacych sposób dzialania tych ukladów;
– specjalizowane uklady wejscia-wyjscia, umozliwiajace podlaczenie specjalnych urzadzen zewnetrznych np.: napedy dysków, monitor ekranowy itp.

W obydwu rodzajach ukladów wejscia–wyjscia mozna wyróznic pewne typowe struktury, takie jak: zespól rejestrów wejsciowych, wyjsciowych, rejestr definiujacy sposób dzialania ukladu oraz rejestr stanu informujacy o aktualnym stanie ukladu wejscia–wyjscia, a którego zawartosc moze byc odczytywana przez mikroprocesor. Uklady wejscia–wyjscia sa wybierane przez mikroprocesor za pomoca adresu. Istnieja dwa sposoby adresowania:

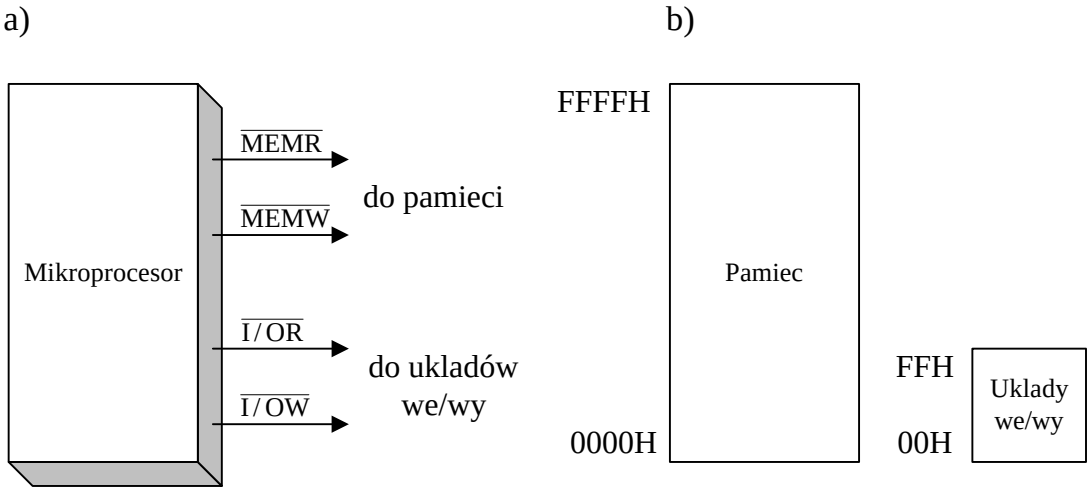
– *adresowanie jednolite* pamieci i ukladów wejscia–wyjscia (ang. *memory mapped I/O*);
– *adresowanie rozdzielne* pamieci i ukladów wejscia–wyjscia (ang. *isolated I/O*).

Adresowanie jednolite (wspólne) polega na dolaczaniu ukladów wejscia–wyjscia w sposób identyczny z modulami pamieci. Pamiec i uklady wejscia–wyjscia nie sa rozrózniane przez mikroprocesor i sa w tej samej przestrzeni adresowej. Dlatego adresy ukladów wejscia–wyjscia musza byc inne niz adresy dolaczonych komórek pamieci. Do sterowania ukladami wejscia–wyjscia nalezy uzywac sygnalów sterujacych pamieci, a w programach do odczytu lub zapisu danych uzywac rozkazów przeslan miedzy pamiecia a rejestrami mikroprocesora. Na rysunku 1.17 wyjasniono zasade jednolitego adresowania. Sygnaly sterujace odczytu pamieci $\overline{\mathbf{MEMR}}$ i zapisu pamieci $\overline{\mathbf{MEMW}}$ bramkowane sygnalem I/O z dekodera adresów, wybierajacego odpowiedni obszar pamieci dla ukladów wejscia–wyjscia tworza sygnaly sterujace zapisem danych $\overline{\mathbf{I/OW}}$ i odczytu danych $\overline{\mathbf{I/OR}}$. Do realizacji przeslania danych uzywa sie w takim przypadku rozkazów komunikacji z pamiecia.



Rys. 1.17. Jednolite adresowanie pamieci i układów wejscia–wyjscia:
a) sygnaly sterujace mikroprocesora, b) mapa pamieci

Przy *adresowaniu rozdzielnym* pamiec i uklady wejscia–wyjscia maja odrebne przestrzenie adresowe. Wówczas adresy układów wejscia–wyjscia i pamieci moga byc takie same. Do sterowania układami uzywa sie sygnałów sterujacych do tego przeznaczonych, a do komunikacji programowej stosuje sie specjalne rozkazy wejscia–wyjscia. Jak przedstawiono na rys. 1.18, sygnaly $\overline{\mathbf{I/OW}}$ i $\overline{\mathbf{I/OR}}$ sa wystawiane bezposrednio przez mikroprocesor i ich pojawienie sie oznacza, ze adres ustawiany na liniach adresowych wskazuje lokalizacje w oddzielnym obszarze adresów wejscia–wyjscia.



Rys. 1.18. Rozdzielne adresowanie pamieci i układów wejscia–wyjscia:
a) sygnaly sterujace mikroprocesora, b) mapa pamieci

Przeslanie danych miedzy mikroprocesorem a ukladami wejscia−wyjscia odbywa sie wówczas tylko przy uzyciu specjalnych rozkazów wejscia−wyjscia. Realizacja tych rozkazów charakteryzuje sie wystepowaniem w odpowiednich cyklach sygnalów $\overline{\mathbf{I/OW}}$ i $\overline{\mathbf{I/OR}}$.

Wyjasnione na rysunkach 1.17 i 1.18 zasady adresowania pamieci i układów wejscia-wyjscia maja charakter ogólny. Zastosowano oznaczenia sygnalów przyjete przez firme INTEL. Nalezy zauwazyc, ze w przypadku mikrokontrolerów jednoukladowych (np. 8051, 8052, SAB80C535), sposób podlaczenia pamieci zewnetrznych i dodatkowych układów wejscia−wyjscia bedzie mial nieco inny charakter. Wynika to z faktu, ze w strukturze mikrokontrolera standardowo znajduja sie uklady wejsc i wyjsc analogowych, cyfrowe uklady wejscia−wyjscia i uklady czasowo−licznikowe. Szczegółowy opis układów wejscia-wyjscia dla mikrokontrolera **SAB 80C535** bedzie przedstawiony w rozdziale drugim.

Wspólpraca mikroprocesora z urzadzeniami zewnetrznymi moze byc organizowana wedlug nastepujacych zasad:

1. Zasady przegladania przez mikroprocesor rejestrów stanów poszczególnych układów wejscia−wyjscia (ang. *pooling*).
2. Zasady realizacji przerwan (ang. *interupt*).
3. Zasady bezposredniego dostepu do pamieci (ang. *Direct Memory Access*).

Wspólpraca na zasadzie przegladania rejestrów stanu odbywa sie calkowicie pod kontrola programu. Rejestr stanu układu wejscia−wyjscia zawiera informacje o aktualnym stanie urzadzenia zewnetrznego. Mikroprocesor przenosi do akumulatora zawartosc rejestru stanu, sprawdza stany odpowiednich bitów i na tej podstawie podejmuje decyzje o realizacji okreslonych dzialan programowych dotyczacych obslugi urzadzenia. Tego typu komunikacja jest bardzo prosta w realizacji sprzetowej, natomiast do jej wad mozna zaliczyc fakt, ze odbywa sie jedynie w scisle okreslonym miejscu programu. Dlatego nie powinna byc stosowana do obslugi urzadzen pracujacych w czasie rzeczywistym (na biezaco).

Wspólpraca z przerwaniami umozliwia praktycznie natychmiastowa reakcje na zadanie obslugi przez urzadzenie zewnetrzne. Istota tego sposobu komunikowania sie

polega na tym, ze mikroprocesor przerywa na chwile wykonywanie aktualnego programu i wykonuje obsluge zglaszajacego sie urzadzenia. Zadanie obslugi jest wprowadzane na *wejscie przerywajace* mikroprocesora za pomoca sygnalu INT (*ang. Interup request*) generowanego przez uklad wejscia–wyjscia w odpowiedzi na sytuacje powstala w urzadzeniu (np. przekroczenie dopuszczalnych wartosci kontrolowanych wielkosci fizycznych). Mikroprocesor sprawdza stan sygnalu INT pod koniec realizacji kazdego rozkazu. Po wykryciu przerwania mikroprocesor wprowadza do rejestru rozkazów zamiast kolejnego kodu rozkazu wskazywanego przez licznik rozkazów – specjalny rozkaz przeznaczony do obslugi przerwania. W wyniku realizacji tego rozkazu mikroprocesor musi:

– wyslac na stos aktualna zawartosc podstawowych rejestrów wewnetrznych (przynajmniej licznika rozkazów), aby po powrocie z programu obslugi przerwania mozna bylo odtworzyc pierwotny stan mikroprocesora i kontynuowac przerwany program,
– wykonac skok do programu obslugi urzadzenia, od którego pochodzi przerwanie.

W ogólnym przypadku mikroprocesor w trakcie realizacji obslugi urzadzenia zewnetrznego powinien:

– zapamietac stan w chwili przyjecia sygnalu przerwania,
– zidentyfikowac zródlo sygnalu przerwania i okreslic jego priorytet, jezeli jednoczesnie pojawila sie wieksza liczba sygnalów przerwan,
– ustalic strategie dzialania obslugi i zrealizowac odpowiedni podprogram,
– odtworzyc stan mikroprocesora i powrócic do przerwanego programu.

Podane zadania moga byc rozwiazywane sprzetowo, programowo lub w sposób mieszany. W przypadku jednoczesnego zgloszenia kilku przerwan, obslugiwane jest przerwanie o najwyzszym priorytecie zwiazanym z okreslona kolejnoscia obslugi. Mozliwe jest równiez wystepowanie wielopoziomowej struktury przerwan, w której podprogram obslugi danego przerwania z jednego urzadzenia moze byc przerwany przez inne przerwania pochodzace z urzadzenia o wyzszym priorytecie drugiego rodzaju.

Przerwania pojawiajace sie w systemie mikroprocesorowym moga byc *maskowane* (przeslaniane). Mozna to realizowac na drodze sprzetowej, stosujac rejestr maski prze-

rwan blokujacy lub przepuszczajacy wybrane odpowiednimi bitami sygnaly przerwan pochodzace od poszczególnych urzadzen wejscia−wyjscia. W przypadku realizacji maskowania na drodze programowej, odpowiedni program sprawdza czy dane przerwanie jest obslugiwane.

Mikroprocesory maja zwykle wiecej niz jedno wejscie przerywajace. Czesc z nich moze byc *niemaskowana*, tzn. niemozliwe jest ich zablokowanie. Pojawienie sie sygnalu przerwania na takim wejsciu powoduje natychmiastowe przerwanie programu i przejscie do odpowiedniego podprogramu obslugi. W ukladach automatyki przemyslowej na wejscia te podawane sa sygnaly wymagajace natychmiastowych dzialan systemu i obslugi.

W trybie *bezposredniego dostepu do pamieci* wymiana danych miedzy pamiecia a urzadzeniami wejscia−wyjscia odbywa sie bez udzialu mikroprocesora. Dzieki temu istnieje mozliwosc szybszego przesylania duzych bloków danych, ograniczonego tylko czasem dostepu do pamieci i czasem dzialania ukladów bioracych udzial w transmisji. Tryb ten jest stosowany przede wszystkim przy wspólpracy z pamieciami dyskowymi i monitorem ekranowym.

1.7. Projektowanie systemów mikroprocesorowych

1.7.1. Etapy projektowania

Kazdy system mikroprocesorowy sklada sie z dwóch scisle za soba powiazanych czesci:

– ***sprzetu*** (ang. *hardware*) czyli mikroprocesora wraz ze wspólpracujacymi ukladami i urzadzeniami,
– ***oprogramowania*** (ang. *software*) czyli zespolu programów umieszczonych w pamieci stalej, realizujacych odpowiednie algorytmy.

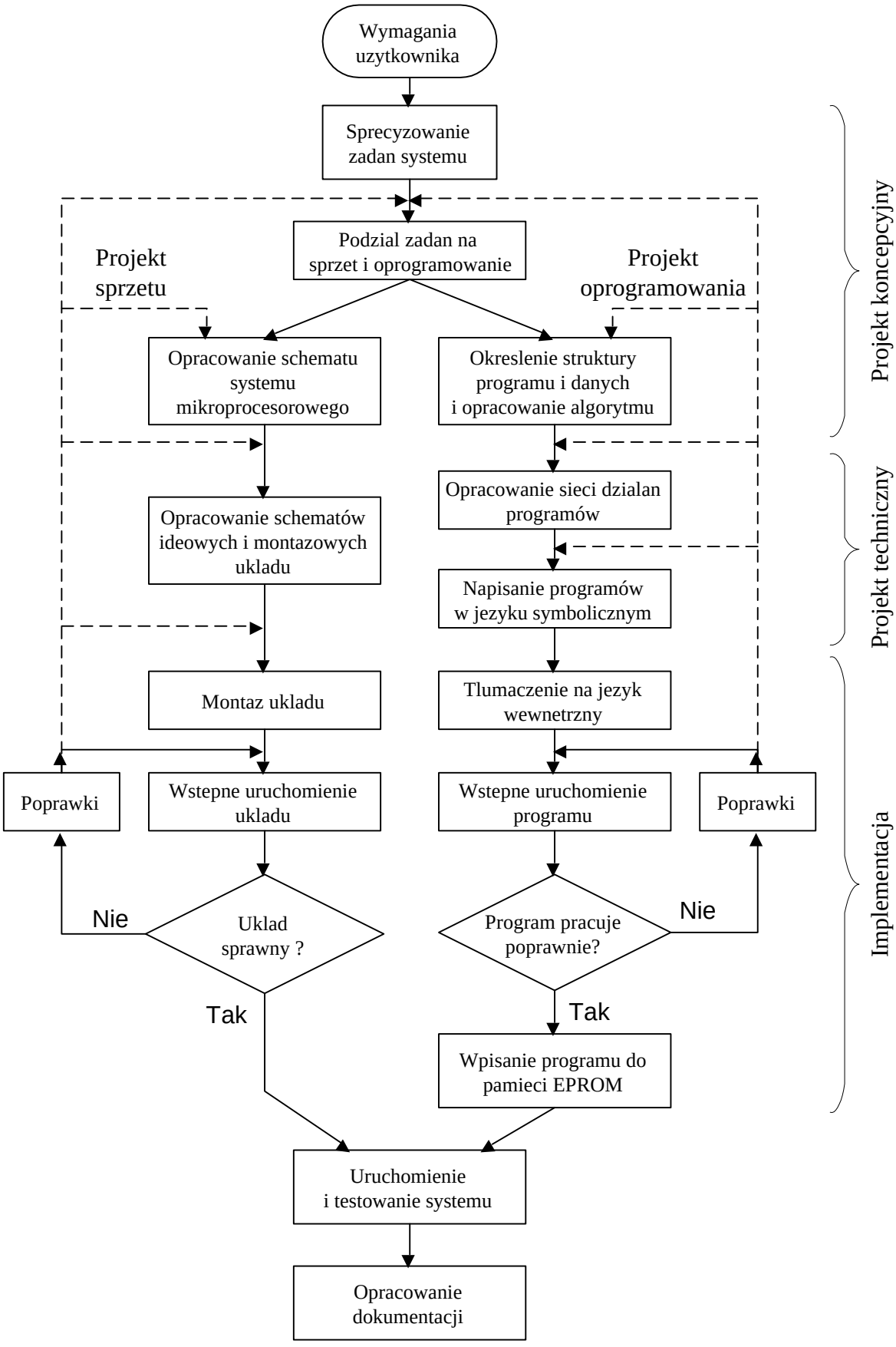
Zaprojektowanie systemu mikroprocesorowego, przeznaczonego np. do sterowania lub kontroli okreslonego obiektu przemyslowego, polega na okresleniu konfiguracji sprzetowej systemu i opracowaniu programów, tak aby postawione na wstepie funkcje i zadania mogly byc zrealizowane. Zwykle koszt opracowania programów jest

duzo wiekszy niz koszt sprzetu.

W ogólnym przypadku proces projektowania specjalizowanego systemu mikroprocesorowego mozna podzielic na nastepujace etapy:

1. ***Projekt koncepcyjny.*** W etapie tym okresla sie kolejno:
   - ogólna koncepcje systemu, czyli sprecyzowanie funkcji i zadan systemu,
   - podzial zadan pomiedzy sprzet i oprogramowanie,
   - schemat funkcjonalny (blokowy) dzialania programu (blokowa struktura programu) i postac danych.
2. ***Projekt techniczny sprzetu i oprogramowania***, w ramach którego nalezy:
   - opracowac schematy ideowe i montazowe mikrokontrolera i ukladów wspólpracujacych,
   - opracowac algorytmy w postaci sieci dzialan programów,
   - napisac programy w jezyku symbolicznym (asembler) lub wysokiego poziomu (np. BASIC lub C).
3. ***Implementacja sprzetu i oprogramowania,*** czyli:
   - montaz ukladu,
   - tlumaczenie programu z postaci symbolicznej na jezyk wewnetrzny i wstepne jego uruchomienie,
   - wpisanie programów do pamieci stalej.
4. ***Uruchomienie systemu***, czyli:
   - polaczenie sprzetu i oprogramowania w jeden dzialajacy poprawnie system,
   - testowanie systemu.
5. ***Opracowanie dokumentacji systemu.***

Na rysunku 1.19 przedstawiono proces projektowania rozdzielajac prace projektowe dotyczace sprzetu i oprogramowania. Po podjeciu decyzji co do podzialu zadan pomiedzy sprzet i oprogramowanie, projektowanie tych dwóch skladników systemu odbywa sie jednoczesnie, ale w scislym powiazaniu ze soba. Scalanie sprzetu i oprogramowania nastepuje dopiero na etapie uruchamiania systemu. Uruchamianie systemu mikroprocesorowego ma na celu wykrycie i usuniecie bledów powstalych we



Rys. 1.19. Proces projektowania systemów mikroprocesorowych

wczesniejszych etapach projektowania sprzetu i programów. Zwykle najwiecej bledów popelnia sie przy pisaniu programów, zwlaszcza w jezyku symbolicznym. Sprzet najczesciej jest oparty na typowych rozwiazaniach i przy projektowaniu systemu mozliwosci popelnienia bledów sa male. W zwiazku z tym glównym problemem w projektowaniu systemów mikroprocesorowych jest uruchomienie programów, zwlaszcza ze dla znalezienia bledów jest konieczne uwazne sledzenie wyników dzialania programu w czasie jego wykonywania.

Sposród wielu metod uruchamiania programów podstawowe znaczenie maja:

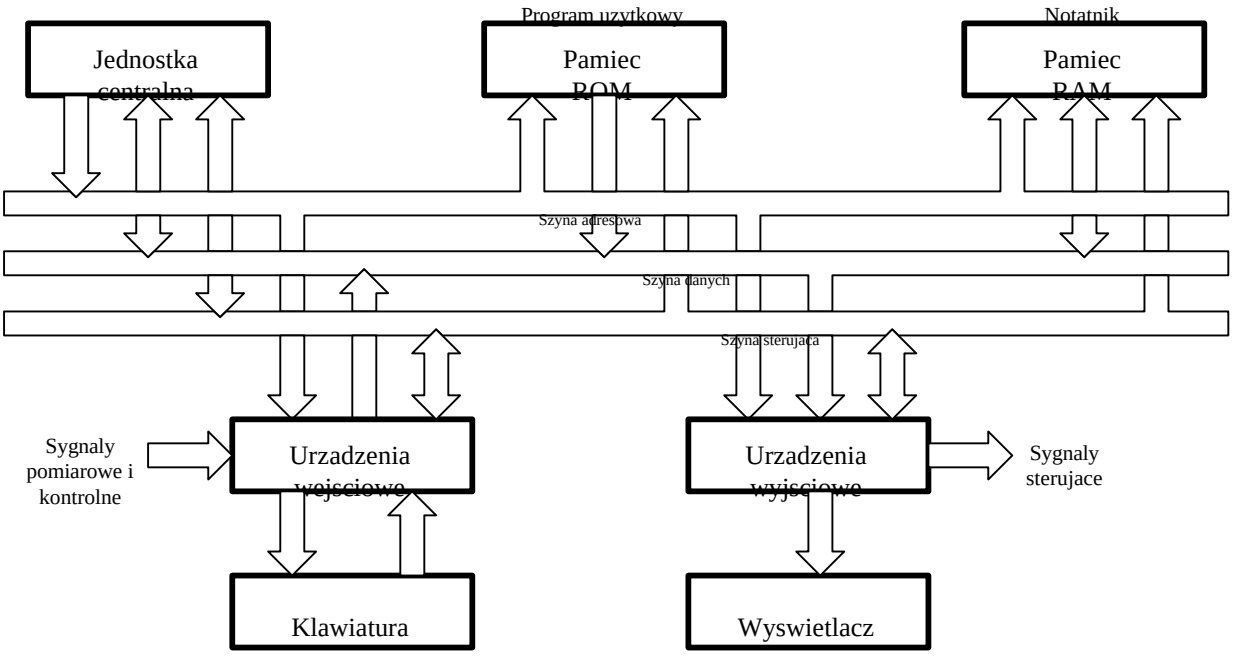
– symulacja ukladu docelowego w duzym komputerze,
– uruchamianie za pomoca specjalnych systemów projektowo-uruchomieniowych.

Komputer realizujacy tlumaczenie programów zródlowych dla mikroprocesora mozna wyposazyc w program symulujacy dzialanie mikroprocesora. Program taki umozliwia przetestowanie przetlumaczonego programu przez symulacje jego wykonywania. Uzytecznosc tej metody jest ograniczona, gdyz testowanie nie obejmuje wspólpracy programu z rzeczywistym obiektem sterowanym przez mikroprocesor.

1.7.2. Systemy uzytkowe i uruchomieniowe

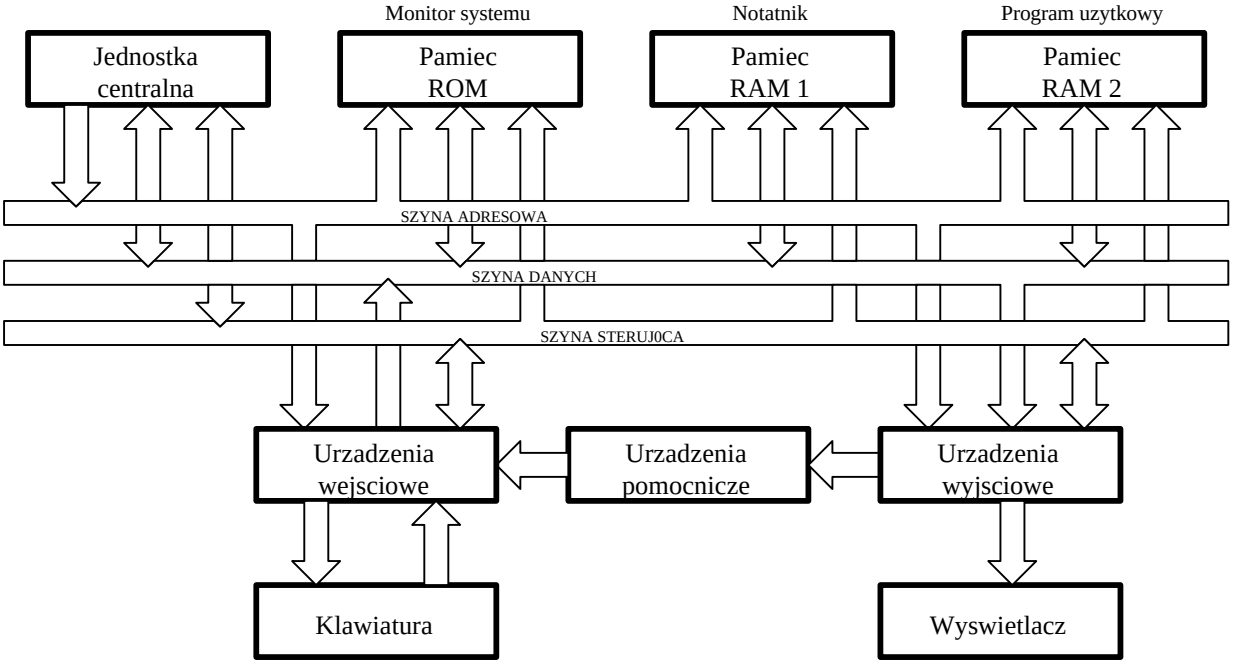
W przypadku duzych systemów komputerowych, zarówno tlumaczenie programów, jak i ich uruchamianie odbywa sie w tym systemie, który docelowo bedzie zainstalowany u uzytkownika. Zwykle producent komputera wyposaza go w dodatkowe oprogramowanie systemowe do tlumaczenia programów i ulatwiajace ich uruchamianie. Natomiast systemy mikroprocesorowe przeznaczone do automatyki przemyslowej sa wbudowane w urzadzenia pomiarowe, kontrolne lub sterujace, stanowiace czesto czesc wiekszego obiektu (urzadzenia). Dlatego nie sa one przystosowane do tlumaczenia programów (nie posiadaja dodatkowej pamieci na przechowywanie programu tlumaczacego i danych ani niezbednych do tego urzadzen zewnetrznych). Równiez uruchamianie programu wykonywanego przez mikroprocesor wbudowany w wyspecjalizowane urzadzenie jest bardzo trudne. Z tego powodu systemy mikroprocesorowe, pod wzgledem sposobu ich wykorzystania, mozna podzielic na ***systemy uzytkowe i uruchomieniowe.***

System uzytkowy (docelowy), czyli taki, jaki bedzie zainstalowany u uzytkownika zawiera sprzet i oprogramowanie przeznaczone tylko do spelnienia funkcji okreslonych w zalozeniach projektu (np. kontrolno-pomiarowych i sterujacych). W zaleznosci od potrzeby, zastepuje on zadajnik, regulatory, przetworniki pomiarowe, sterowniki cyfrowe itp. Uruchamianie w nim programu byloby bardzo trudne ze wzgledu na to, ze bezposrednio mozna obserwowac tylko zewnetrzne wyniki dzialania procesora oraz nie ma mozliwosci dokonywania zmian w programie. ***Program uzytkowy***, realizowany przez mikroprocesor, jest przechowywany w pamieci stalej *ROM*. Dane zmienne znajduja sie w pamieci zapisywalnej *RAM* tworzac tzw. notatnik. Sygnaly pomiarowe i kontrolne sa wprowadzane za posrednictwem urzadzen wejsciowych, a sygnaly sterujace obiektem sa wyprowadzane za posrednictwem urzadzen wyjsciowych. Uzytkownik kontaktuje sie z systemem za pomoca klawiatury i wyswietlacza. W systemie uzytkowym nie ma miejsca na umieszczenie specjalnego oprogramowania i urzadzen zewnetrznych wspomagajacych i ulatwiajacych uruchamianie. Mozliwa jest oczywiscie rozbudowa systemu, ale wiaze sie to ze znacznym zwiekszeniem jego ceny. Na rysunku 1.20 przedstawiono schemat blokowy mikroprocesorowego systemu uzytkowego.



Rys. 1.20. Schemat blokowy systemu uzytkowego

System uruchomieniowy (rys. 1.21) sluzy do przygotowywania programów uzytkowych dla systemów uzytkowych (docelowych).



Rys. 1.21 Schemat blokowy systemu uruchomieniowego

Porównujac ogólne schematy blokowe systemów uzytkowego i uruchomieniowego, widac powtarzanie sie zasadniczych podzespolów. Jednakze wystepuja znaczne róznice w ich wykorzystaniu, a czesciowo równiez w budowie. W systemie uruchomieniowym program uzytkowy miesci sie w dodatkowej pamieci zapisywalnej *RAM* o pojemnosci kilku tysiecy bajtów. W pamieci stalej *ROM* jest zapisany program zarzadzajacy, tzw. *monitor systemu*, umozliwiajacy tworzenie programu uzytkowego przez wprowadzanie kolejnych rozkazów z klawiatury do pamieci programu uzytkowego *RAM* oraz nadzorujacy wykonywanie innych programów (redagujacego, tlumaczacego i testujacego). W systemie uruchomieniowym bardziej jest rozbudowana klawiatura, która najczesciej ma postac pelnej klawiatury alfanumerycznej. Blok urzadzen pomocniczych obejmuje najczesciej monitor ekranowy, zewnetrzna pamiec, drukarke oraz urzadzenia do programowania pamieci stalych (*PROM*, *EPROM*). Dlatego tez w systemie uruchomieniowym zupelnie inaczej sa skonstruowane urzadzenia wejscia i wyjscia. Po

wyposazeniu w odpowiednie sprzegi, system uruchomieniowy moze oczywiscie spelniac takze funkcje systemu uzytkowego. Obecnie równiez czesto stosuje sie specjalne systemy uruchomieniowe umozliwiajace tzw. e*mulacje ukladowa*. Polega ona na tym, ze po wstepnym uruchomieniu programu uzytkowego system uruchomieniowy przez specjalny uklad sprzegajacy zostaje wlaczony w miejsce mikroprocesora ukladu docelowego. Tym samym procesor ukladu uruchomieniowego steruje rzeczywistymi urzadzeniami zewnetrznymi ukladu docelowego, umozliwiajac jednoczesnie sledzenie wykonywania programu uzytkowego.

1.7.3. Programowanie systemu mikroprocesorowego

Jezyki programowania, sluzace do zapisu programów wykonywanych w mikroprocesorze, dziela sie na ***wewnetrzne, symboliczne*** i ***wyzszego poziomu (proceduralne).*** *Jezyk wewnetrzny* mikroprocesora jest jedynym jezykiem zrozumialym dla jego ukladu sterowania. Zapis programu w tym jezyku stanowia ciagi zer i jedynek (slowo binarne). Jest on malo czytelny dla czlowieka i dlatego tez do zapisu programów stosuje sie *jezyki symboliczne.* Program zapisany w jezyku symbolicznym jest programem zródlowym. W wyniku tlumaczenia tego programu na postac binarna (wewnetrzna) otrzymuje sie tzw. *program wynikowy.* Proces tlumaczenia nazywa sie *translacja*.

Najprostszym jezykiem symbolicznym jest *jezyk asemblera*, w którym jednej instrukcji odpowiada jeden rozkaz mikroprocesora. Instrukcje jezyka asemblera odzwierciedlaja konstrukcje mikroprocesora oraz jego liste rozkazów. Podstawowe cechy jezyka asemblera to:

- stosowanie symbolicznych kodów rozkazów mikroprocesora w postaci skrótów mnemonicznych,
- stosowanie adresów symbolicznych.

Od podanych cech pochodzi nazwa *jezyk symboliczny*. Instrukcje jezyka symbolicznego mozna podzielic na:

- instrukcje wlasciwe, tj. instrukcje tlumaczone na jeden rozkaz mikroprocesora,
- pseudoinstrukcje sluzace do sterowania procesem tlumaczenia programu zródlowego

w jezyku asemblera na jezyk wewnetrzny mikroprocesora oraz do generowania danych,

– makroinstrukcje, tj. instrukcje tlumaczone na ciag rozkazów mikroprocesora, sluzace do generowania czesto spotykanych w programie sekwencji rozkazów.

Jezyki symboliczne posiadajace mozliwosc definiowania makroinstrukcji nazywa sie *jezykami makroasemblera.*

Linia programu zródlowego stanowi jedna instrukcje. Instrukcja sklada sie z czterech czesci, zwanych polami: etykiety, rozkazu (mnemonika rozkazu), operandów (argumentów), komentarza.

Pole etykiety zawiera nazwe, zaczynajaca sie od litery i zakonczona dwukropkiem. Etykieta jest zazwyczaj symboliczna nazwa komórki pamieci (adresem symbolicznym), w której jest umieszczony dany rozkaz. Pole etykiety moze byc puste. *Pole rozkazu* zawiera symboliczna nazwe rozkazu, a wiec okresla rodzaj operacji. *Pole operandów*, zaleznie od typu rozkazu, zawiera jeden lub dwa operandy rozdzielone przecinkiem; moze tez ono byc puste. *Komentarz* jest dowolnym tekstem nastepujacym po sredniku.

Informacja zapisywana jest w polu operandów; zaleznie od zawartosci pola rozkazu, moze okreslac rejestr procesora, adres pamieci lub operand bezposredni. Adres lub operand bezposredni moze byc:

– liczba binarna, heksadecymalna lub dziesietna,

– symbolem (nazwa), któremu nadano wartosc przez umieszczenie go w polu etykiety innej instrukcji lub za pomoca specjalnej dyrektywy.

Przyklad instrukcji jezyka asemblera:

```
ALFA:   MOV R1,#201 ; wpisz liczbe 201 do rejestru R1
Etykieta Rozkaz Argumenty          Komentarz
```

Tlumaczenie programu napisanego w jezyku asemblera na postac binarna, polega na przypisaniu kazdemu symbolowi (nazwie rozkazu, rejestru, adresowi symbolicznemu) wartosci odpowiednich bitów slów rozkazowych. Dla wykonania tego zadania translator asemblera potrzebuje dodatkowych informacji mówiacych, w jakim obszarze pamieci ma byc umieszczony program i dane.

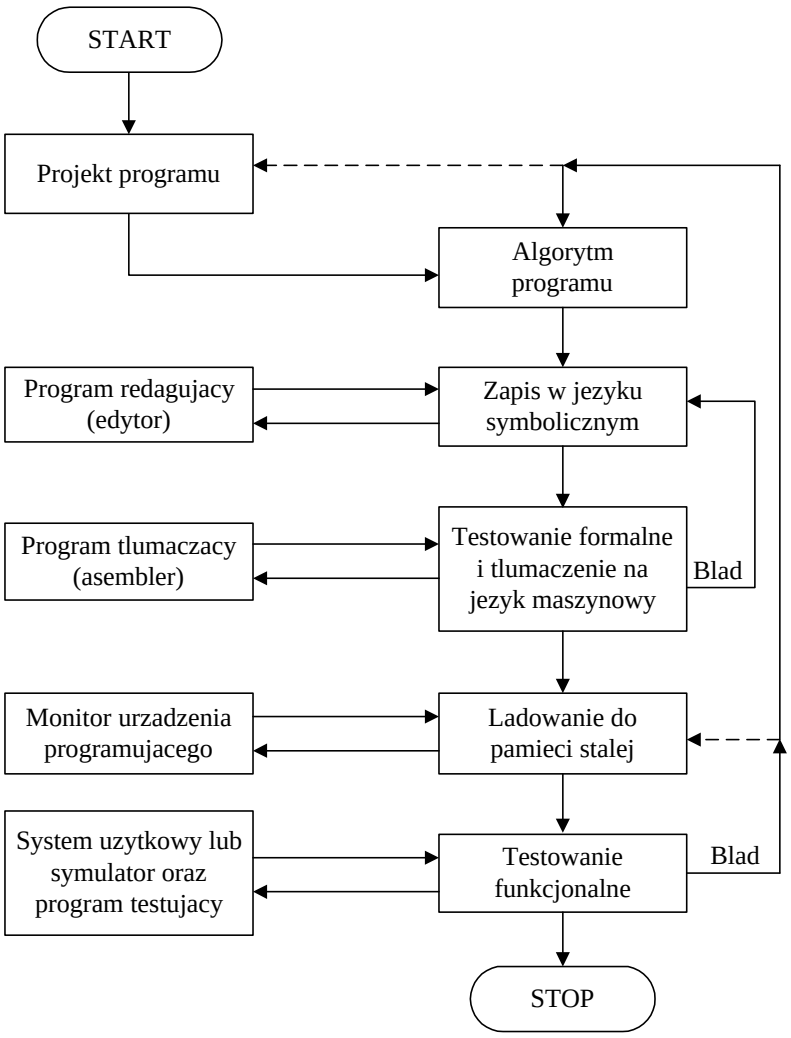
Programowanie w jezyku asemblera umozliwia pisanie programów optymalnych ze wzgledu na zajetosc pamieci lub czas wykonywania, ale jest stosunkowo trudne i pracochlonne. Listy rozkazów mikroprocesorów róznia sie od siebie, a wiec temu samemu algorytmowi odpowiadaja rózne programy dla róznych mikroprocesorów. Dlatego tez coraz czesciej programuje sie mikrokontrolery w *jezykach wyzszego poziomu* niezaleznych od listy rozkazów mikroprocesora i stosujacych symbolike stosowana w arytmetyce. Tlumaczenie programu zródlowego napisanego w takim jezyku jest duzo bardziej skomplikowane niz tlumaczenie programu napisanego w jezyku asemblera. Tlumaczenie to dokonuje program zwany *kompilatorem*. Jezeli kazdej instrukcji asemblera odpowiada jeden rozkaz programu wynikowego, to jednej instrukcji jezyka wyzszego poziomu odpowiada ciag rozkazów mikroprocesora.

Typowymi przykladami jezyków wyzszego poziomu sa jezyki BASIC i C. Sa one stosowane przede wszystkim do mikroprocesorów za slowem maszynowym 16–bitowym i wiecej oraz procesorów sygnalowych i nie beda one omawiane w niniejszej pracy.

Przy opracowywaniu duzego programu niezbedne jest zachowanie dyscypliny i systematycznosci. W szczególnosci duzy program nalezy podzielic na mniejsze czesci, realizujace okreslone funkcje i dajace sie samodzielnie uruchamiac. Czesci takie moga byc realizowane jako podprogramy. Przyklad procesu tworzenia programu uzytkowego przedstawiono na rys. 1.22.

Wstepny projekt programu uscisla sie stopniowo, np. korzystajac z sieci dzialan (ang. *flow diagram*), az do uzyskania algorytmu programu. Algorytm ten zapisuje sie w jezyku symbolicznym za pomoca programu redagujacego (tzw. *edytora*). Program tlumaczacy umozliwia sprawdzenie zapisu pod wzgledem formalnym oraz przeklada zapis na jezyk maszynowy. W przypadku wykrycia bledów poprawia sie zapis w wykonany w jezyku symbolicznym. Sprawdzony program uzytkownika jest zapisywany w pamieci stalej i sprawdzany pod wzgledem funkcjonalnym w systemie uzytkowym lub za pomoca symulatora obiektu. W przypadku stwierdzenia nieprawidlowosci dzialania, której nie mozna poprawic przy uzyciu programu testujacego, konieczne jest poprawienie algorytmu lub ewentualnie zmiana projektu koncepcyjnego, po czym caly

proces nalezy powtórzyc.



Rys. 1.22. Proces tworzenia programu uzytkowego

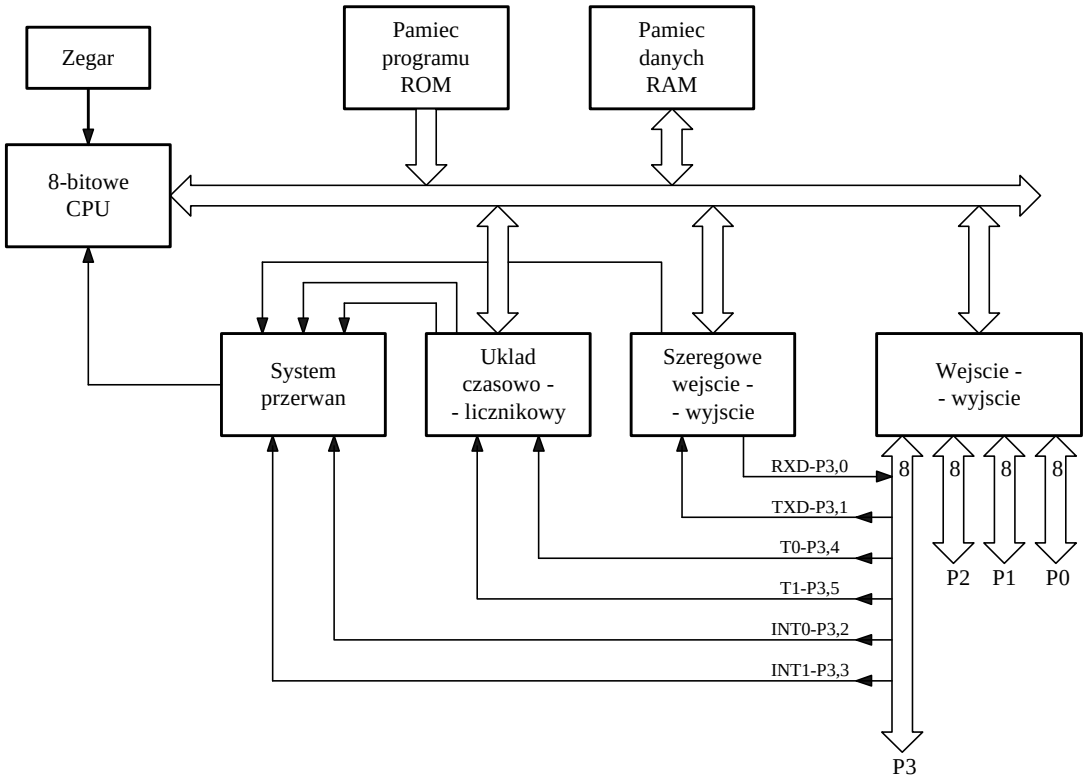
Uruchamianie i testowanie programu wynikowego ulatwia program uruchomieniowy (ang. *debugger*), który umozliwia krokowe wykonywanie uruchamianego programu, zatrzymywanie jego dzialania w wybranych punktach (pulapkach), wyswietlanie na monitorze i modyfikacje zawartosci rejestrów i komórek pamieci.

1.8. Charakterystyka rodziny mikroprocesorów INTEL MCS-51

Architektura mikroprocesorów rodziny INTEL MCS-51 jest podobna do wczesniejszej rodziny mikrokontrolerów jednoukladowych tej firmy MCS-48. Sa one jed-

nakze znacznie rozbudowane i unowoczesniane. Podstawowymi ukladami rodziny MCS-51 sa: mikrokontroler 8051, od którego pochodzi nazwa rodziny oraz nieco rozbudowany mikrokontroler 8052. Funkcjonalny schemat blokowy jednoukladowego mikrokontrolera 8051 i podstawowego w rodzinie MCS-51, przedstawiono na rys.1.23.

Osmiobitowa jednostka centralna moze wykonywac 111 rozkazów (49 jedno–, 45 dwu–, i 17 trzybajtowych), umozliwiajacych latwa i efektywna realizacje wszelkiego rodzaju algorytmów sterowania i zlozonych obliczen. Lista rozkazów zawiera m.in. rozkazy arytmetyczne (w tym mnozenie i dzielenie) i logiczne, rozkazy dotyczace operacji logicznych na bitach oraz rozbudowane grupy rozkazów skoków warunkowych i wejscia–wyjscia. Prawie wszystkie rozkazy wykonuja sie w czasie jednego lub dwóch cykli maszynowych.



Rys. 1.23. Schemat funkcjonalny mikrokontrolera 8051

Wyjatek stanowi tu mnozenie i dzielenie, wymagajace czterech cykli. Zegar taktujacy jest stabilizowany zewnetrznym rezonatorem kwarcowym o czestotliwosci maksymalnej 12 MHz. Ze wzgledu na wewnetrzny dzielnik 1/12 czas cyklu maszynowego

jest równy 1 μs. Wewnetrzna pamiec programu ROM ma pojemnosc 4 kB. Moze byc rozszerzona do 64 kB przez dolaczenie pamieci zewnetrznej. Wewnetrzna pamiec danych RAM ma pojemnosc 128 bajtów. Mozliwe jest dolaczenie zewnetrznej pamieci danych o pojemnosci do 64 kB (w ramach osobnej przestrzeni adresowej). Uklad czasowo–licznikowy zawiera dwa 16–bitowe liczniki, które moga zliczac wewnetrzne impulsy zegarowe lub impulsy zewnetrzne. Oba liczniki moga pracowac w jednym z czterech, ustawianych indywidualnie trybów.

Linie wejscia–wyjscia, których jest 32, sa zorganizowane w cztery 8–bitowe porty. Czesc z tych linii moze byc wykorzystana do realizacji specjalnych funkcji.

Port szeregowy umozliwia niezalezne nadawanie i odbieranie transmisji szeregowej. Moze pracowac w czterech trybach.

Uklad przerwan (dwupoziomowy) moze obslugiwac dwa przerwania zewnetrze i dwa z ukladu czasowo-licznikowego oraz przerwanie z ukladu szeregowego wejscia-wyjscia.

Mikrokomputer 8052 jest wersja ukladu 8051 o nieco powiekszonych zasobach, a mianowicie:

– wewnetrzna pamiec programu jest powiekszona do 8 kB,
– wewnetrzna pamiec danych jest powiekszona do 256–bajtów,
– uklad czasowo-licznikowy zawiera dodatkowy 16–bitowy licznik impulsów zegarowych lub zewnetrznych.

W ramach rodziny MCS-51 sa produkowane mikrokontrolery rózne pod wzgledem technologii wykonania i rodzaju wewnetrznej pamieci programu (np. 80C51, 80C31, 8051AH, 8751H, 8031AH, 8032). Mikrokontrolery rodziny MCS-51sa obecnie najpopularniejszymi ukladami na swiecie. Takie firmy jak SIEMENS, Signetics /Philips, AMD, Fujitsu i inne, produkuja odpowiedniki ukladów firmy INTEL. Zazwyczaj sa one oznaczane takim samym symbolem cyfrowym. Ponadto, wiele firm produkuje inne, wzorowane na 8051, wersje ukladów, rózniace sie technologia wykonania lub rozbudowane o dodatkowe specjalizowane bloki funkcjonalne, przeznaczone do konkretnego zastosowania przemyslowego.