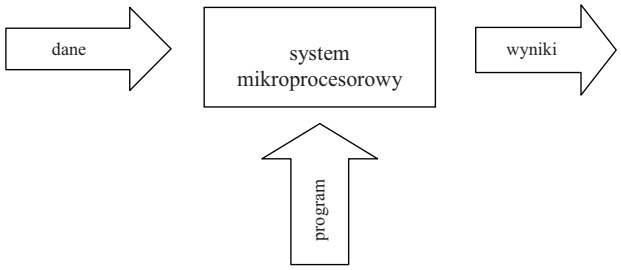
4. MATERIAŁ NAUCZANIA

4.1. System mikroprocesorowy

4.1.1. Materiał nauczania

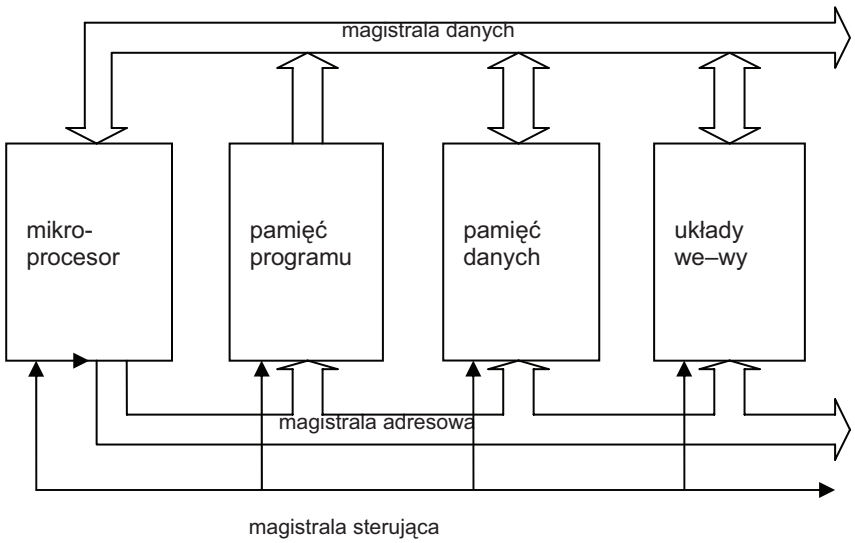
Systemy mikroprocesorowe służą do przetwarzania informacji. Przetwarzanie informacji polega na dostarczeniu do systemu danych, które poddawane są określonym działaniom dając wyniki. Wynikami mogą być sygnały sterujące pracą maszyn, obrazy, teksty itp. Jedną z ważniejszych części tego systemu jest procesor, który przetwarza informacje, wykonując na niej elementarne operacje zwane instrukcjami (bądź rozkazami). Ciąg takich instrukcji realizujący konkretne zadanie przetwarzania informacji nazywamy programem. Tak więc do systemu mikroprocesorowego musimy dostarczyć dane wejściowe, program lub zestaw programów, aby po przetworzeniu otrzymać wynik.



Rys. 1. System mikroprocesorowy

Typowy system mikroprocesorowy składa się z następujących części:
- mikroprocesora
- pamięci danych RAM
- pamięci programu ROM
- układów wejścia/wyjścia
- układów sterujących przepływem informacji między tymi elementami: magistrali danych, magistrali adresowej i sygnałów sterujących.

Schemat blokowy systemu mikroprocesorowego opartego na architekturze von Neumana:



Rys. 2. Schemat systemu mikroprocesorowego

Mikroprocesor pełniący funkcję jednostki centralnej umożliwia wykonywanie operacji przetwarzania danych poprzez realizację programu zapisanego w pamięci programu.

Mikroprocesor składa się z dwóch układów: układu sterowania oraz jednostki arytmetyczno-logicznej wraz z rejestrami roboczymi.

Zadaniem układu sterowania jest pobieranie rozkazów z pamięci programu, dekodowanie ich oraz wystawianie odpowiednich sygnałów sterujących (wewnętrznych i zewnętrznych) w celu wykonania rozkazów.

Jednostka arytmetyczno-logiczna służy do wykonywania operacji arytmetycznych lub logicznych na liczbach binarnych. Jednostka arytmetyczno logiczna zawiera również rejestr wskaźników (flagowy), w którym ustawiane są znaczniki wyniku wykonanej operacji (np. flaga znaku, zera, przeniesienia).

Pamięć programu – pamięć nieulotna, przechowuje program w języku maszynowym. Najczęściej jest wykonywana jako pamięć typu:

–ROM – programowana przez producenta
–PROM – programowana jednorazowo przez użytkownika
–EPROM, EEPROM – do wielokrotnego programowania przez użytkownika

Pamięć danych – pamięć typu odczyt/zapis, służy do przechowywania danych podczas realizacji programu.

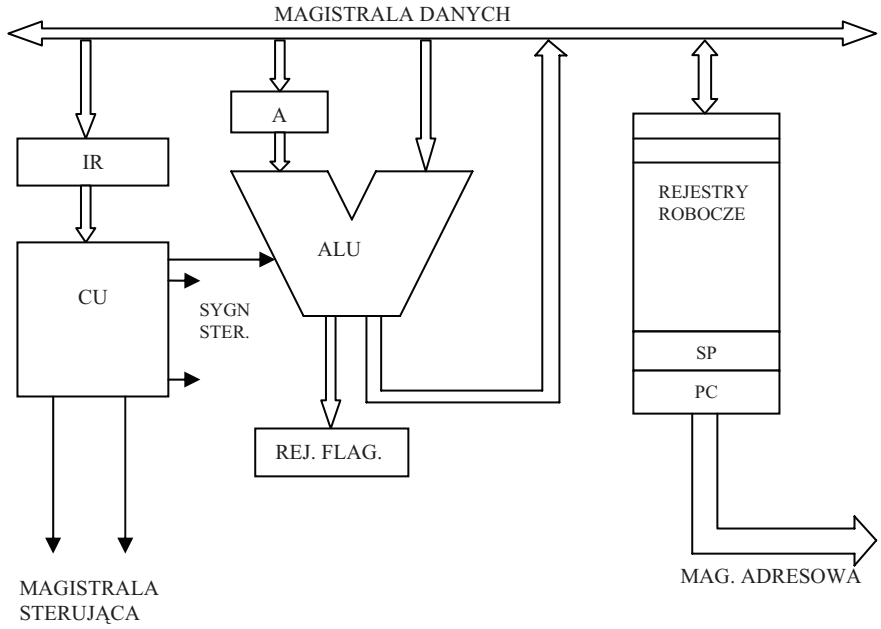
Układy wejścia/wyjścia – umożliwiają prawidłową komunikację między mikroprocesorem a otoczeniem – urządzeniami zewnętrznymi. Zadaniem układów wejścia/wyjścia jest zapewnienie odpowiedniej postaci danej (np. zamiana z postaci szeregowej na równoległą i odwrotnie) oraz dopasowanie czasowe wymiany danej (np. zapamiętanie danej do czasu, kiedy odbierze ją urządzenie zewnętrzne).

Magistrala danych – zespół linii, którymi przesyłane są liczby binarne (dane, kody rozkazów, słowa sterujące i statusowe).

Magistrala adresowa – zespół linii, którymi procesor adresuje poszczególne komórki pamięci programu, danych lub rejestry układów wejścia/wyjścia.

Magistrala sterująca – zespół linii, którymi wymieniane są sygnały sterujące.

Mikroprocesor zawiera następujące podzespoły:
- jednostkę arytmetyczno–logiczną ALU (arithmetic and logic unit),
- akumulator A (accumulator),
- licznik rozkazów PC (program counter),
- zestaw rejestrów roboczych R_i (general purpose register),
- rejestr rozkazów IR (instruction register),
- wskaźnik stosu SP (stack pointer),
- układ sterowania CU (control unit).



Rys. 3. Schemat blokowy mikroprocesora

Jednostka arytmetyczno–logiczna (ALU) wykonuje podstawowe działania logiczne i arytmetyczne na słowach takich jak suma logiczna, iloczyn logiczny, suma modulo 2, negacja, dodawanie i odejmowanie arytmetyczne, porównywanie, rotacja. Wykonanie operacji arytmetycznej lub logicznej powoduje ustawienie znaczników (flag) w rejestrze wskaźników. Najczęściej stosowane flagi to:
- przeniesienia **CY** (przyjmuje wartość 1 jeżeli wystąpiło przeniesienie z najbardziej znaczącej pozycji),
- przeniesienia połówkowego **AC** (przyjmuje wartość 1 jeżeli wystąpiło przeniesienie z bitu 3 na 4 – wykorzystywany w operacjach na liczbach BCD),
- zera **Z** (przyjmuje wartość 1 jeżeli wynik operacji jest równy 0),
- znaku **S** (przyjmuje wartość 1 jeżeli wynik operacji jest ujemny),
- parzystości **P** (przyjmuje wartość 1 jeżeli liczba jedynek w wyniku operacji jest nieparzysta).

Akumulator – rejestr związany z jednostką arytmetyczną – z akumulatora jest domyślnie pobierany jeden z argumentów operacji dwuargumentowej, w akumulatorze jest umieszczany wynik operacji. Akumulator jest również wykorzystywany podczas komunikacji z urządzeniami zewnętrznymi.

Rejestry robocze mogą pełnić różną role, w zależności od realizowanego rozkazu. Mogą służyć do przechowywania danych lub adresów. Na rejestrach można wykonywać niektóre operacje wynikające z listy rozkazów.

Licznik rozkazów PC służy do adresowania pamięci programu tzn. przechowuje adres komórki pamięci, z której należy pobrać kod następnego rozkazu. W czasie realizacji programu licznik zwiększa swoją zawartość po pobraniu każdego rozkazu lub jest do niego wpisywana wartość (np. adres skoku).

Rejestr rozkazów IR służy do przechowywania pobranego z pamięci programu kodu rozkazu.

Układ sterowania CU dekoduje zawartość rejestru rozkazów i generuje sygnały sterujące zapewniające realizację pobranego rozkazu.

Wskaźnik stosu SP służy do adresowania wydzielonego obszaru pamięci danych, w którym zapisywane są adresy i dane zgodnie z regułą LIFO (last in first out) tzn. kolejność odczytu słowa jest odwrotna do kolejności ich zapisywania. Tak jest zorganizowana programowa realizacja stosu, niektóre mikroprocesory posiadają stos sprzętowy. Stos i wskaźnik stosu umożliwiają przejście do podprogramu, a po jego zakończeniu powrót do programu głównego.

Mikroprocesory są układami sekwencyjnymi synchronicznymi, dlatego też z układem sterowania musi współpracować generator impulsów zegarowych.

Rozkazy są realizowane w cyklach rozkazowych. Każdy cykl maszynowy składa się z cykli maszynowych. Cykl maszynowy to część cyklu rozkazowego związana z odwołaniem się (w celu odczytu lub zapisu) do pamięci lub układu wejścia/wyjścia. Cykl maszynowy składa się z określonej liczby taktów zegarowych.

Każdy cykl rozkazowy można podzielić na dwie fazy:
- faza pobrania rozkazu z pamięci programu,
- faza wykonania rozkazu, która ma inny przebieg dla każdego rozkazu.

Faza pobrania rozkazu jest jednakowa dla wszystkich rozkazów. Polega na pobraniu kodu rozkazu z pamięci programu, z komórki której adres wskazuje zawartość licznika rozkazów PC. Kod pobranego rozkazu jest umieszczany w rejestrze rozkazów IR, licznik rozkazów zostaje zwiększony, tak aby wskazywał następny rozkaz.

Faza wykonania rozkazu ma różny przebieg, rozpoczyna się w chwili umieszczenia rozkazu w rejestrze rozkazów IR. Układ sterowania CU na podstawie zdekodowanego stanu rejestru generuje odpowiednią sekwencję sygnałów sterujących zapewniających prawidłową realizację rozkazu.

Podczas realizacji programu argumenty mogą być pobierane/zapisywane z/do rejestrów roboczych, pamięci danych, pamięci programu (tylko odczytywane) lub układów wejścia/wyjścia. Decyduje o tym użyty sposób adresacji. W mikroprocesorach stosowane są:
- adresacja natychmiastowa – rozkaz zawiera kod rozkazu i argument (tzn. argument jest przechowywany wraz z kodem rozkazu w pamięci programu)
- adresacja bezpośrednia – rozkaz zawiera kod rozkazu i adres argumentu w pamięci danych
- adresacja rejestrowa – rozkaz zawiera kod rozkazu i nazwę rejestru, w którym przechowywany jest argument
- adresacja rejestrowa pośrednia – rozkaz zawiera kod rozkazu i nazwę rejestru, w którym przechowywany jest adres argumentu w pamięci danych.

4.1.2. Pytania sprawdzające

Odpowiadając na pytania, sprawdzisz, czy jesteś przygotowany do wykonania ćwiczeń.

1. Z jakich elementów składa się system mikroprocesorowy?
2. Jakie jest przeznaczenie poszczególnych bloków?
3. Z jakich podzespołów składa się mikroprocesor?
4. Do czego służy akumulator?
5. Jakie jest przeznaczenie rejestrów roboczych?
6. Co to jest stos i do czego służy?
7. W jaki sposób jest adresowany stos?
8. Co to jest cykl rozkazowy, cykl maszynowy, takt zegarowy?
9. W jakich fazach odbywa się cykl rozkazowy?
10. Do czego służy licznik rozkazów, kiedy zmieniana jest jego zawartość?
11. Co to jest rejestr flagowy (wskaźników), co oznaczają poszczególne flagi?
12. Jakie są metody adresacji stosowane w mikrokontrolerach?

4.1.3. Ćwiczenia

Ćwiczenie 1

Połącz elementy z lewej strony z ich opisem po prawej stronie. Przeanalizuj zadanie dla mikroprocesora ośmiobitowego (8–bitowa magistrala danych) z 16 bitową magistralą adresową.

akumulator	16– bitowy rejestr adresowy pamięci programu
rejestr roboczy	8–bitowa magistrala dwukierunkowa
licznik rozkazów	wydzielony fragment pamięci danych
magistrala danych	8– bitowa magistrala jednokierunkowa
wskaźnik stosu	8– bitowy rejestr, w którym umieszczany jest wynik operacji arytmetyczno-logicznej
adresacja natychmiastowa	8– bitowy rejestr ogólnego przeznaczenia
stos	argument rozkazu jest pobierany z pamięci danych
magistrala adresowa	argument rozkazu jest pobierany z pamięci programu
adresacja rejestrowa pośrednia	16– bitowy rejestr adresowy pamięci danych
	16– bitowa jednokierunkowa magistrala

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) przeanalizować podane informacje,
2) dopasować opisy do odpowiednich elementów systemu mikroprocesorowego lub metod adresacji (jeden opis pozostanie niewykorzystany),
3) zaprezentować wykonane ćwiczenie,
4) dokonać oceny poprawności wykonanego ćwiczenia.

Wyposażenie stanowiska pracy:
- treść zadania dla każdego ucznia,
- zeszyt przedmiotowy,
- literatura z rozdziału 6.

Ćwiczenie 2

W jednostce arytmetyczno logicznej odbywają się wymienione operacje. Uzupełnij tabelę, określając jaki będzie 8–bitowy wynik działania (binarnie), czyli zawartość akumulatora po wykonaniu operacji oraz jakie będą flagi po ich wykonaniu: wpisz 0– jeżeli flaga będzie zerowana, 1– jeśli będzie ustawiana lub x – jeśli wykonanie operacji nie wpływa na flagę. Dane w tabeli są dziesiętne, pamiętaj, że jednostka arytmetyczno-logiczna wykonuje operacje na danych binarnych, musisz więc dokonać konwersji:

operacja	wynik operacji w akumulatorze	C	Z	P
135+100				
167+200				
$115 \oplus 115$				
$120 \wedge 100$				
$15 \wedge 240$				
$255 \vee 15$				

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) dokonać konwersji danych dziesiętnych na binarne,
2) wykonać podane działania,
3) na podstawie uzyskanych wyników uzupełnić tabelę,
4) zaprezentować wykonane ćwiczenie,
5) dokonać oceny poprawności wykonanego ćwiczenia.

Wyposażenie stanowiska pracy:
- treść zadania dla każdego ucznia,
- zeszyt przedmiotowy,
- literatura z rozdziału 6.

4.1.4. Sprawdzian postępów

Czy potrafisz:

		Tak	Nie
1)	rozpoznać poszczególne elementy systemu mikroprocesorowego na podstawie schematu blokowego?	☐	☐
2)	opisać rolę elementów systemu mikroprocesorowego?	☐	☐
3)	wymienić podzespoły mikroprocesora i podać ich funkcję?	☐	☐
4)	wyjaśnić pojęcia cykl rozkazowy, cykl maszynowy, takt zegarowy?	☐	☐
5)	wyjaśnić z jakich faz składa się cykl rozkazowy?	☐	☐
6)	opisać metody adresacji?	☐	☐
7)	przyporządkować metodę adresacji do lokalizacji argumentu?	☐	☐

4.2. Komunikacja mikroprocesora z otoczeniem

4.2.1. Materiał nauczania

Zastosowanie systemu mikroprocesorowego do realizacji konkretnych zadań wymaga zapewnienia wymiany danych między mikroprocesorem i urządzeniami zewnętrznymi (klawiatura, wyświetlacz, czujniki, układy wykonawcze), a w niektórych przypadkach także między pamięcią danych a urządzeniami zewnętrznymi.

Układy wejścia/wyjścia pośredniczą w wymianie informacji między mikroprocesorem a urządzeniami zewnętrznymi. Układy wejścia/wyjścia można podzielić na:

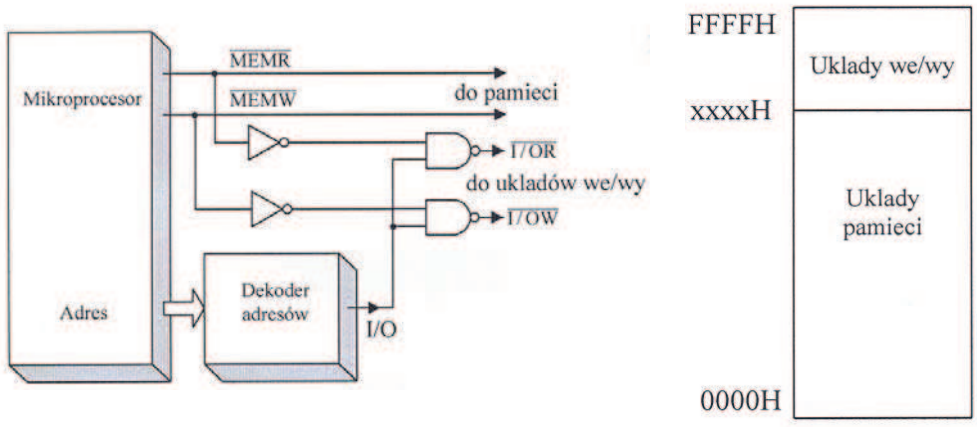
- proste układy wejścia/wyjścia, w których podstawowym elementem jest rejestr danych lub bramka trójstanowa,
- programowalne, uniwersalne układy wejścia/wyjścia, w których oprócz rejestrów danych występują rejestry sterujące pozwalające na ustalenie określonego trybu pracy układu, rejestr statusowy, przechowujący informację o stanie układu oraz wewnętrzny układ sterowania
- specjalizowane układy wejścia/wyjścia służące do współpracy z konkretnymi urządzeniami

Komunikacja mikroprocesora z otoczeniem odbywa się za pośrednictwem magistral. Układ jest wybierany odpowiednim adresem.

Wyróżnia się dwie metody adresowania pamięci i układów wejścia/wyjścia:

- adresowanie jednolite pamięci i układów wejścia/wyjścia (układy wejścia/wyjścia współadresowalne z pamięcią),
- adresowanie rozdzielone pamięci i układów wejścia/wyjścia.

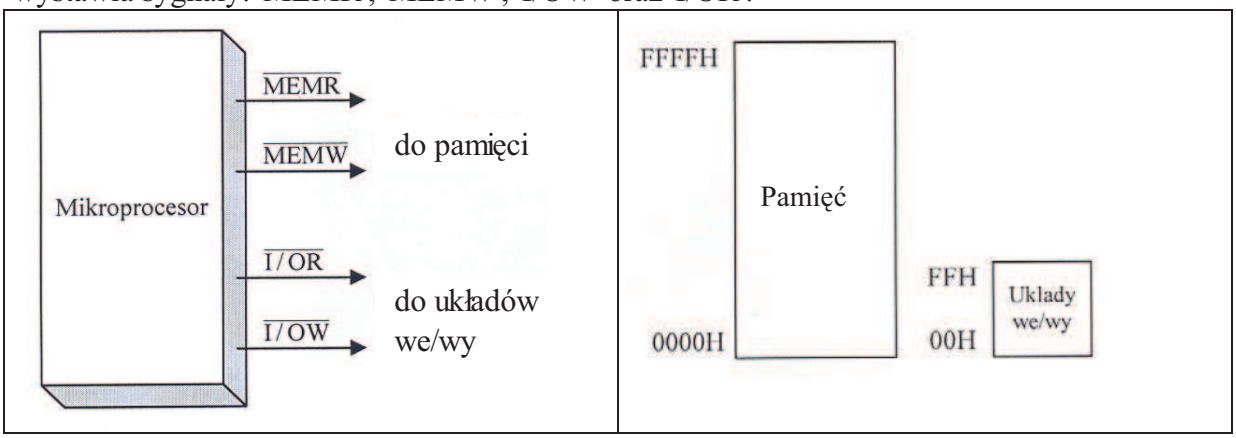
Adresowanie jednolite (wspólne) polega na dołączaniu układów we/wy w taki sam sposób jak modułów pamięci. Pamięć i układy we/wy nie są rozróżniane przez procesor i znajdują się w tej samej przestrzeni adresowej. Te same rozkazy i te same sygnały sterujące służą do komunikacji z pamięcią i z układami we/wy. Dlatego układy we/wy muszą mieć przydzielone inne adresy niż komórki pamięci.



Rys. 4. Jednolite adresowanie pamięci i układów wejścia/ wyjścia [1, str39]

Sygnały sterujące odczytu pamięci $\overline{MEMR}$ i zapisu pamięci $\overline{MEMW}$ są bramkowane sygnałem I/O z dekodera adresów wybierającego odpowiedni obszar pamięci dla układów we/wy i tworzą sygnały sterujące zapisem danych $\overline{I/OW}$ i odczytu danych $\overline{I/OR}$.

Adresowanie rozdzielone polega na tym, że pamięć i układy we/wy mają odrębne przestrzenie adresowe – adresy komórek pamięci i układów we/wy mogą być takie same. Lista rozkazów zawiera odrębną grupę rozkazów dotyczącą komunikacji z pamięcią, a odrębną do komunikacji z układami wejścia–wyjścia. Mikroprocesor bezpośrednio wystawia sygnały: $\overline{MEMR}$, $\overline{MEMW}$, $\overline{I/OW}$ oraz $\overline{I/OR}$.



Rys. 5. Rozdzielone adresowanie pamięci i układów wejścia/ wyjścia [1, str39]

Sposoby komunikacji między mikroprocesorem a otoczeniem:
- obsługa programowa (pooling)
- przerwania (interrupt)
- bezpośredni dostęp do pamięci (Direct Memory Access)

Obsługa programowa polega na przeglądaniu przez mikroprocesor stanów poszczególnych układów we/wy. Mikroprocesor w sposób programowy pobiera zawartości rejestrów statusowych poszczególnych układów we/wy, sprawdza stan odpowiednich bitów i na ich podstawie podejmuje decyzje dotyczące określonych działań programowych dotyczących obsługi urządzenia. Tego typu komunikacja jest bardzo prosta w realizacji sprzętowej, jej wadą jest fakt, że odbywa się w ściśle określonym momencie realizacji programu. Nie może więc być stosowana do obsługi urządzeń pracujących w czasie rzeczywistym, gdzie konieczna jest natychmiastowa obsługa dołączonych urządzeń.

Współpraca z przerwaniami umożliwia praktycznie natychmiastową reakcję na żądanie obsługi przez urządzenie zewnętrzne. W momencie przyjęcia zgłoszenia żądania obsługi przez urządzenie procesor zawiesza realizację programu na czas obsługi urządzenia. Układ we/wy generuje sygnał żądania przerwania INT, mikroprocesor sprawdza stan tego sygnału pod koniec realizacji każdego rozkazu. Po wykryciu przerwania mikroprocesor:
- zapamiętuje na stosie stan licznika rozkazów (adres rozkazu, który byłby realizowany, gdyby przerwanie nie nadeszło),
- wpisuje do licznika rozkazów adres podprogramu obsługi przerwania (wektor przerwania),
- realizuje podprogram obsługi przerwania,
- po zakończeniu obsługi przerwania przywraca ze stosu stan licznika rozkazów i wraca do realizacji programu głównego.

Jeżeli zgłasza się więcej niż jedno przerwanie układ przerwań rozstrzyga, które z przerwań ma być obsłużone na podstawie priorytetu.

Układ przerwań może być jedno– lub wielopoziomowy. W jednopoziomowym układzie przerwań nie ma możliwości zawieszenia realizowanego podprogramu obsługi przerwania przez żadne inne przerwanie (niezależnie od priorytetu). W wielopoziomowym układzie przerwań, przerwanie o wyższym poziomie może przerwać podprogram obsługi przerwania o niższym poziomie. Podprogram obsługi przerwania o niższym poziomie zostanie dokończony po zrealizowaniu obsługi przerwania o wyższym poziomie.

Przerwanie może zostać przyjęte do obsługi, jeżeli:
- procesor zakończył realizację rozkazu,
- zgłaszające się przerwanie jest odblokowane,
- nie zgłasza się inne przerwanie o priorytecie wyższym,
- nie jest realizowana obsługa innego przerwania (w układzie przerwań jednopoziomowym) lub przerwania o równym lub wyższym poziomie (w układzie przerwań wielopoziomowym)

Tryb bezpośredniego dostępu do pamięci DMA jest stosowany podczas wymiany danych między pamięcią a urządzeniami zewnętrznymi bez udziału mikroprocesora. Dzięki temu możliwe jest szybsze przesyłanie bloków danych. Przesyłanie odbywa się pod kontrolą sterownika DMA, mikroprocesor w tym czasie nie ma dostępu do magistral, nie realizuje więc swojego programu. Przed rozpoczęciem transmisji DMA sterownik musi zostać zaprogramowany przez umieszczenie w jego rejestrach sterujących informacji dotyczących kierunku transmisji, ilości przesyłanych słów i adresu obszaru danych. Po zakończeniu przesyłu następuje powrót do realizowanego programu.

4.2.2. Pytania sprawdzające

Odpowiadając pytania, sprawdzisz, czy jesteś przygotowany do wykonania ćwiczeń.
1. Jaką rolę pełnią układy wejścia/wyjścia?
2. Jakie są rodzaje układów wejścia/wyjścia?
3. Jakie są metody adresowania?
4. Jakie są sposoby współpracy mikroprocesora z otoczeniem?
5. Na czym polega obsługa programowa?
6. O czym podczas obsługi przerwań decyduje priorytet, a o czym poziom?
7. Na czym polega obsługa z przerwaniami?
8. Na czym polega bezpośredni dostęp do pamięci?

4.2.3. Ćwiczenia

Ćwiczenie 1

Wybierz, które zdania są prawdziwe, a które fałszywe:

Zdanie:	prawda	fałsz
Adresowanie jednolite pozwala na zaadresowanie większej przestrzeni adresowej niż adresacja rozdzielona.		
Przy zastosowaniu adresacji jednolitej te same rozkazy odnoszą się do pamięci i układów wejścia/wyjścia.		
Układ wejścia/wyjścia umożliwia dopasowanie czasowe między szybszym mikroprocesorem, a wolniejszym urządzeniem		

wejścia/wyjścia.		
W przypadku jednoczesnego zgłoszenia się dwóch zgłoszeń przerwań do obsługi zostanie przyjęte przerwanie o wyższym poziomie.		
Obsługa programowa nie pozwala na ustalanie priorytetów przerwań.		
Po zakończeniu obsługi przerwania program główny jest wykonywany od rozkazu, w czasie którego nadeszło przerwanie.		
Przerwanie może być przyjęte do obsługi pod warunkiem zakończenia realizowanego cyklu rozkazowego.		
Podczas bezpośredniego dostępu do pamięci dane są przesyłane z mikroprocesora do pamięci danych.		

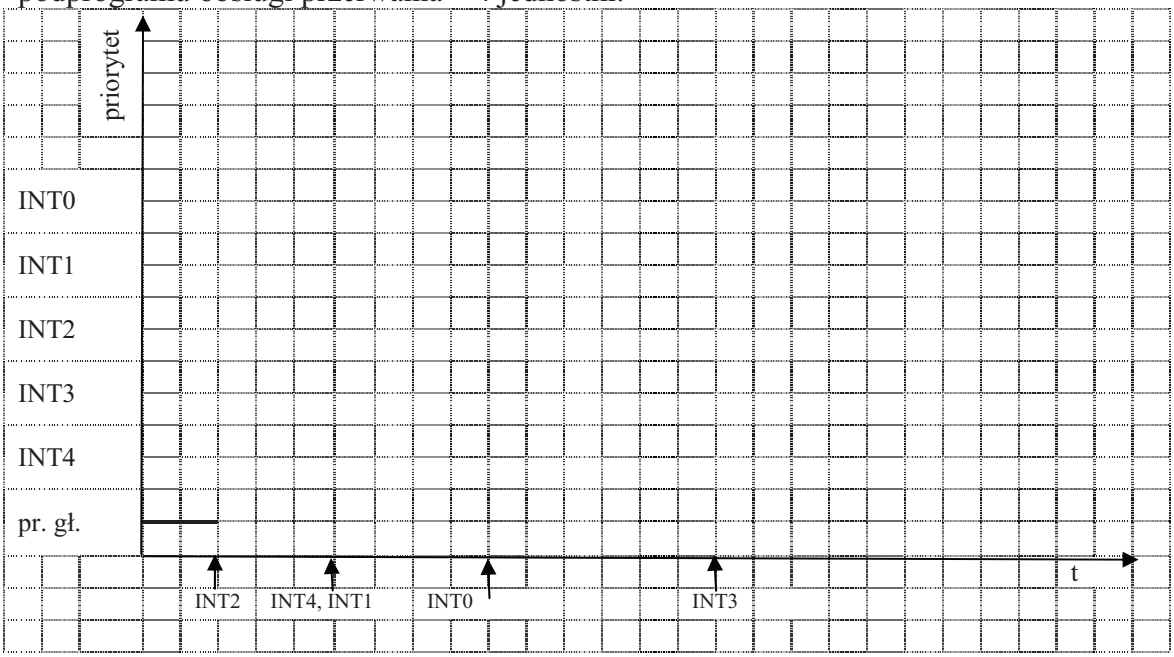
Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zapoznać się ze sposobami komunikacji mikroprocesora z otoczeniem,
2) dokładnie przeczytać zdania,
3) przeanalizować ich treść decydując, czy zdanie jest prawdziwe czy fałszywe,
4) zaprezentować wykonane ćwiczenie,
5) dokonać oceny poprawności wykonanego ćwiczenia.

Wyposażenie stanowiska pracy:
- treść zadania dla każdego ucznia,
- literatura z rozdziału 6.

Ćwiczenie 2

Przedstaw na wykresie kolejność obsługi przerwań w jednopoziomowym układzie przerwań. Wszystkie przerwania są odblokowane, wszystkie mogą zostać przyjęte do obsługi. Przerwanie INT0 ma najwyższy priorytet, a INT4 najniższy. Czas trwania każdego podprogramu obsługi przerwania – 4 jednostki.



Na tym samym wykresie innym kolorem narysuj kolejność obsługi przerwań w wielopoziomowym układzie przerwań, przyjmując, że przerwanie INT0 ma poziom najwyższy i dalej każde kolejne ma poziom niższy, INT4 ma poziom najniższy.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) w przypadku jednopoziomowego układu przerwań rozstrzygnąć, które ze zgłoszeń ma w danej chwili najwyższy priorytet,
2) w przypadku wielopoziomowego układu przerwań dodatkowo zdecydować, czy możliwe jest przejście do obsługi przerwania podczas trwania obsługi realizowanego podprogramu,
3) narysować przebiegi kolejności obsługi przerwań,
4) zaprezentować wykonane ćwiczenie,
5) dokonać oceny poprawności i estetyki wykonanego ćwiczenia.

Wyposażenie stanowiska pracy:
- treść zadania dla każdego ucznia,
- literatura z rozdziału 6.

4.2.4. Sprawdzian postępów

Czy potrafisz:

		Tak	**Nie**
1)	rozróżnić metody adresacji: jednolitą i rozdzieloną?	☐	☐
2)	wyjaśnić jakie wady i zalety ma każda z metod?	☐	☐
3)	scharakteryzować role układów wejścia/wyjścia?	☐	☐
4)	rozróżnić rodzaje układów wejścia/wyjścia?	☐	☐
5)	scharakteryzować sposoby komunikacji mikroprocesora z otoczeniem?	☐	☐
6)	wyjaśnić, kiedy korzystne jest stosowanie każdej z tych metod?	☐	☐
7)	wymienić warunki przyjęcia przerwania?	☐	☐
8)	opisać działania mikroprocesora po przyjęciu przerwania?	☐	☐
9)	zdefiniować pojęcia poziom, priorytet, wektor przerwania?	☐	☐

4.3. Mikrokontrolery rodziny ‘51

4.3.1.Materiał nauczania

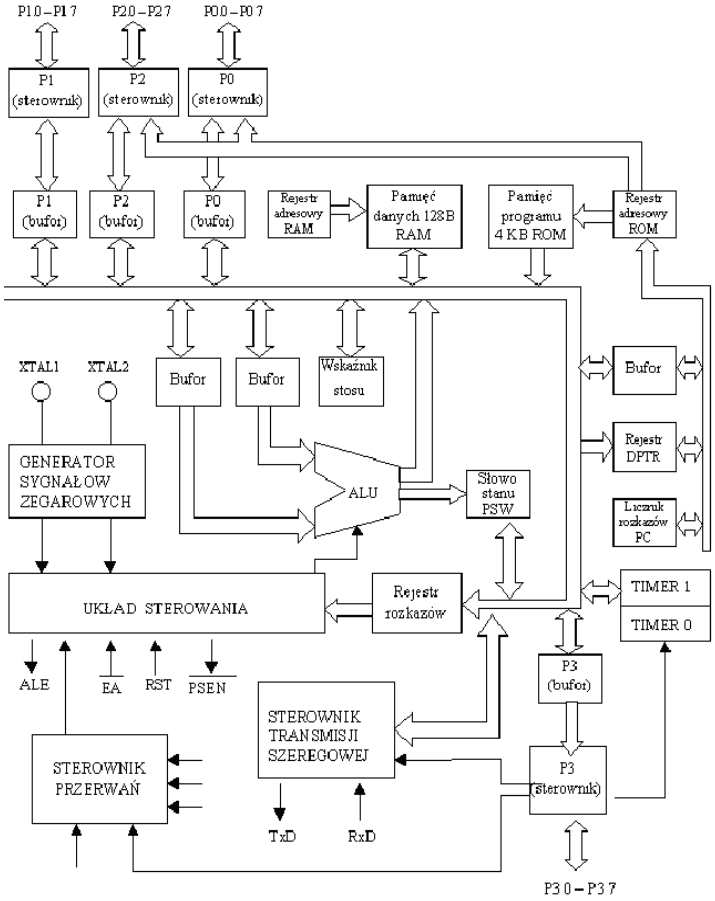
Mikrokomputerami jednoukładowymi, zwanymi również mikrokontrolerami lub mikrosterownikami, nazywana jest pewna klasa mikroprocesorów charakteryzująca się tym, że w jednym układzie scalonym zawarty jest mikroprocesor, pamięć danych, układy we/wy umożliwiające bezpośrednie dołączenie urządzeń zewnętrznych, układy czasowo licznikowe, układ przerwań i inne zasoby, w niektórych mikrokontrolerach jest wbudowana pamięć programu.

Charaktrystyka rodziny mikrokontrolerów ’51:

- 8–bitowa magistrala danych,
- 16–bitowa magistrala adresowa,
- 8–bitowy arytmometr,
- 128 B (256B) wewnętrznej pamięci danych,
- dwa 16–bitowe układy czasowo–licznikowe,
- cztery 8–bitowe porty równoległe,
- układ transmisji szeregowej,
- układ przerwań,
- możliwość dołączenia zewnętrznej pamięci danych,
- możliwość dołączenia zewnętrznej pamięci danych,

Częstotliwość pracy mikroprocesorów zawiera się w granicach od 1,2 do 20 MHz.

Architektura podstawowego mikrokontrolera rodziny ’51:



Rys. 6. Schemat mikrokontrolera 8051 [7]

ALU – jednostka arytmetyczno–logiczna
PSW – rejestr statusowy

C	AC	F0	RS1	RS0	OV	–	P

znaczenie flag:
C – przeniesienie (z bitu 7)
AC – przeniesienie połówkowe (z bitu 3 na 4)
F0 – znacznik ogólnego przeznaczenia, ustawiany i kasowany programowo
RS1, RS0 – wybór bieżącego banku rejestrów roboczych
OV – znacznik nadmiaru
P – flaga parzystości

Rejestry:
A, ACC – rejestr akumulatora,
B– rejestr dodatkowy używany przy mnożeniu i dzieleniu,
DPTR – rejestr adresowy zewnętrznej pamięci,
PC– licznik programu,
SP– wskaźnik stosu,
P0–P3– porty równoległe
linie portu **P0 i P2** mogą służyć do dołączenia pamięci zewnętrznych,
alternatywne funkcje portu P3 przedstawia tabela 1.

Tabela 1 Alternatywne funkcje portu P3

linia	Funkcja
P3.0	RxD– we portu szeregowego
P3.1	TxD– wy portu szeregowego
P3.2	INT0– we przerwania zewnętrznego
P3.3	INT1– we przerwania zewnętrznego
P3.4	T0– we licznika 0
P3.5	T1– we licznika 1
P3.6	WR– sygnał zapisu do zewnętrznej pamięci
P3.7	RD– sygnał odczytu z zewnętrznej pamięci

Sygnały sterujące:
$\overline{\text{PSEN}}$ – sygnał odczytu z zewnętrznej pamięci programu,
ALE – sygnał zatrzaskujący w zewnętrznym rejestrze młodszą część adresu podczas współpracy z zewnętrznymi pamięciami,
$\overline{\text{EA}}$ – sygnał wyboru pamięci programu ($\overline{\text{EA}}$=0– odczyt z zewnętrznej pamięci programu, $\overline{\text{EA}}$=1 z wewnętrznej),
RST – sygnał zerujący procesor , zerowany jest licznik PC (start programu od komórki zerowej), nie jest zerowana pamięć RAM.

Pamięć wewnętrzna **IRAM**
Tabela 2 Przestrzeń adresowa pamięci IRAM

Adresy 48–127 (30H–7FH)	Pamięć danych użytkownika
Adresy 32–47 (20H–2FH)	Pamięć adresowana bitowo (adresy 0–127 (0H–7FH))
Adresy 24–31 (18H–1FH)	Rejestry R0–R7 – bank 3 (RB3)
Adresy 16–23 (10H–17H)	Rejestry R0–R7 – bank 2 (RB2)

Adresy 8–15 (8H–0FH)	Rejestry R0–R7 – bank 1 (RB1)
Adresy 0–7 (0H–7H)	Rejestry R0–R7 – bank 0 (RB0)

Obszar od adresu 0 do 31 (0H–1FH) zajmują cztery banki rejestrów (RB0...RB3) roboczych, po osiem rejestrów w banku. Rejestry te mają oznaczenia R0 do R7 i mogą być wykorzystywane do przechowywania danych. Wyjątek stanowią rejestry R0 i R1 każdego bloku, które mogą być dodatkowo wykorzystane do indeksowego adresowania wewnętrznej i zewnętrznej pamięci danych. W danej chwili dostępny jest tylko bank wybierany bitami RS1 i RS0 rejestru statusowego PSW. W obszar e pamięci o adresach 32–47(20H–2FH) możliwe jest zaadresowanie pojedynczego bitu komórki pamięci. Bity te są dostępne pod adresami 0–127 (0H–7FH). Obszar pamięci o adresach 48–127 (30H–7FH) nie posiada już żadnych specyficznych własności i wykorzystywany jest jak zwykła pamięć o organizacji bajtowej. Cała pamięć może być adresowana jak pamięć o bajtowej organizacji w sposób bezpośredni lub rejestrowy pośredni używając rejestrów R0 i R1.

Rejestry specjalne SFR:

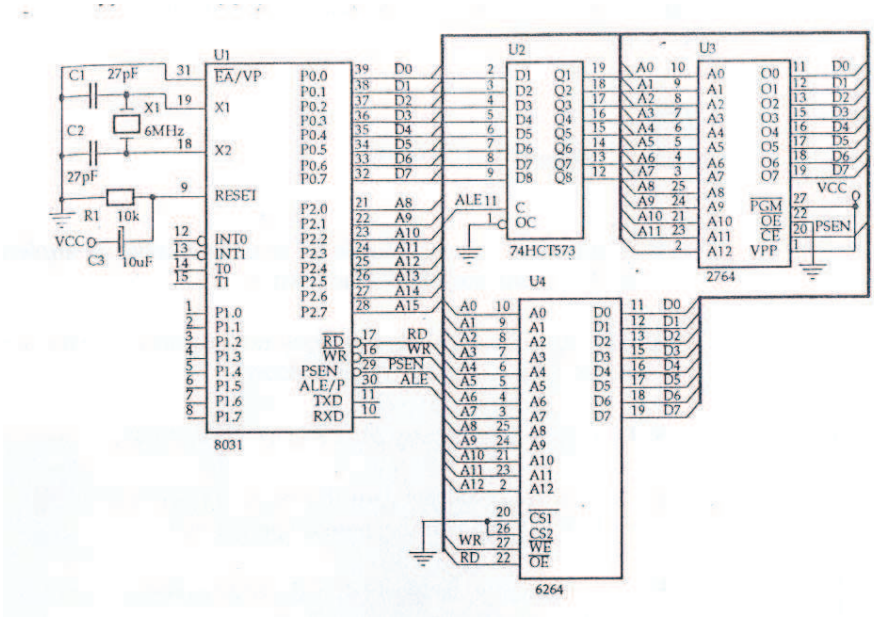
Blok rejestrów specjalnych (SFR; ang. Special Function Registers) znajduje się w niespójnym obszarze pamięci danych mikrokontrolera o adresach 128–240 (80H–0F0H). Obszar rejestrów SFR mikrokontrolera '51 jest wykorzystywany dwojako – z jednej strony umieszczone są w nim wszystkie (za wyjątkiem licznika rozkazów i czterech banków rejestrów R0–R7) rejestry sterujące pracą mikrokontrolera lub wykorzystywane bezpośrednio przy wykonywaniu programu; z drugiej zaś strony rejestry SFR stanowią rodzaj interfejsu pomiędzy mikroprocesorem a układami peryferyjnymi umieszczonymi wewnątrz mikrokontrolera. Wszystkie operacje sterowania wewnętrznymi układami peryferyjnymi oraz przesyłania danych między nimi a CPU, odbywają się właśnie za pośrednictwem rejestrów SFR. Dostęp do każdego z tych rejestrów możliwy jest wyłącznie w trybie adresowania bezpośredniego. Nazwy rejestrów SFR są zazwyczaj nazwami predefiniowanymi, więc najczęściej nie trzeba znać adresu danego rejestru – wystarczy pamiętać jego nazwę.

Tabela 3 Obszar SFR

Nazwa	Adres	Pełniona funkcja
P0	128(80H)	Port we/wy 0
SP	129(81H)	Wskaźnik stosu
DPL	130(82H)	Rejestr indeksowy DPTR (mniej znaczący bajt)
DPH	131(83H)	Rejestr indeksowy DPTR (bardziej znaczący bajt)
PCON	135(87H)	Rejestr sterujący stanami uśpienia
TCON	136(88H)	Rejestr sterujący układów czasowych 0 i 1
TMOD	137(89H)	Rejestr trybu pracy układów czasowych 0 i 1
TL0	138(8AH)	Rejestr danych układu czasowego 0 (mniej znaczący)
TL1	139(8BH)	Rejestr danych układu czasowego 1 (mniej znaczący)
TH0	140(8CH)	Rejestr danych układu czasowego 0 (bardziej znaczący)
TH1	141(8DH)	Rejestr danych układu czasowego 1 (bardziej znaczący)
P1	144(90H)	Port we/wy 1
SCON	152(98H)	Rejestr sterujący układu transmisji szeregowej
SBUF	153(99H)	Rejestr danych układu transmisji szeregowej
P2	160(0A0H)	Port we/wy 2
SP	129(81H)	Wskaźnik stosu
IE	168(0A8H)	Rejestr maski przerwań
P3	176(0B0H)	Port we/wy 3
IP	184(0B8H)	Rejestr priorytetów przerwań
PSW	208(0D0H)	Słowo stanu procesora

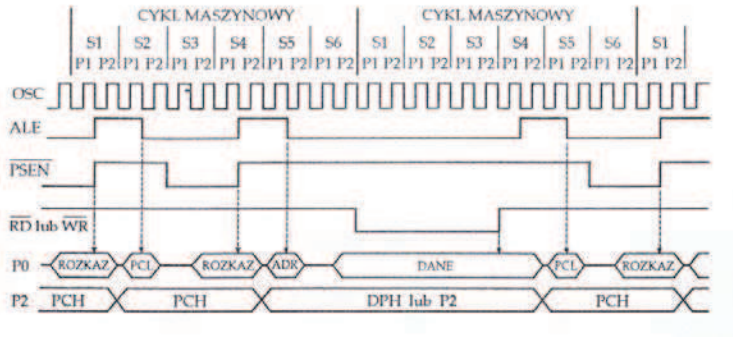
Pamięć zewnętrzna

Oprócz pamięci wewnętrznej danych możliwe jest dołączenie do mikrokontrolera zewnętrznej pamięci danych o pojemności do 64kB. Dostęp do tego obszaru pamięci jest tylko używając adresacji rejestrowej pośredniej, adres znajduje się w rejestrach DPTR, R0 lub R1. Jeżeli do adresacji użyto 16–bitowego rejestru DPTR, zawartość DPH jest wpisywana do portu P2, a DPL do P0. Jeżeli do adresacji użyto R0 lub R1 modyfikowana jest tylko zawartość P0.



Rys. 7. Dołączenie zewnętrznej pamięci danych i programu [2, s.45]

Przy odczycie lub zapisie danej do pamięci zewnętrznej po odczytaniu rozkazu w takcie S1, w takcie S5 następuje wysłanie na wyjście portu P0 mniej znaczącego bajtu adresu, który w drugiej fazie tego taktu jest zatrzaskiwany w rejestrze sygnałem ALE. Na wyjście portu P2 jest podawany starszy bajt adresu. W pierwszej fazie pierwszego taktu S1P1 drugiego cyklu maszynowego wykonywanie operacji dostępu do XRAM, jeden z sygnałów $\overline{WR}$ lub $\overline{RD}$ (w zależności od kierunku przesyłania danej) przechodzi w stan niski uaktywniając pamięć. Następuje odczyt lub zapis, który kończy się w pierwszej fazie czwartego taktu S4P1.



Rys. 8. Sygnały sterujące i zegarowe przy dostępie do zewnętrznej pamięci danych [2, s.44]

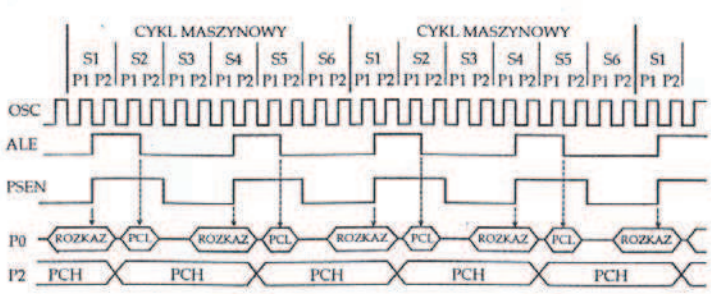
Pamięć programu

Obszar pamięci programu o maksymalnej objętości 64 kB może składać się z dwóch części:

- pamięci wewnętrznej o pojemności 4kB lub 8kB,
- pamięci zewnętrznej.

Jeżeli mikrokontroler posiada wewnętrzną pamięć programu to zmieniając stan sygnału $\overline{EA}$ można wybierać między pamięciami wewnętrzną ($\overline{EA}$=1 i adres mieszczący się w wewnętrznej pamięci programu) i zewnętrzną ($\overline{EA}$=1 i adres przekraczający obszar wewnętrznej pamięci programu lub $\overline{EA}$=0). Sposób dołączenia pamięci zewnętrznej programu został przedstawiony na rysunku 7.

Przy pobieraniu rozkazów i danych z zewnętrznej pamięci programu jest generowany sygnał PSEN. Podczas pobierania rozkazu z pamięci zewnętrznej w takcie S5 przez port P0 wysyłany jest mniej znaczący bajt adresu. W drugiej fazie tego taktu (S5P2) sygnał ALE zatrzaskuje ten bajt w rejestrze. Równocześnie na wyjście portu P2 jest wysyłany starszy bajt adresu. W pierwszej fazie szóstego taktu (S6P1) sygnał PSEN zmieniając swój stan na niski uaktywnia pamięć programu i zostaje wystawiony bajt rozkazu.



Rys. 9. Sygnały sterujące i zegarowe przy dostępie do zewnętrznej pamięci programu [2, s.43]

Adresowanie różnych typów pamięci

Ponieważ mikrokontroler nie posiada osobnych rozkazów do dostępu do urządzeń wejścia/wyjścia, to w obszarze adresowym 64kB zewnętrznej pamięci danych mogą być również umieszczane rejestry dołączanych do systemu urządzeń we/wy.

Ponadto adresy wewnętrznej pamięci danych RAM pokrywają się z adresami zewnętrznej pamięci danych RAM i pamięci programu, zatem, aby rozróżnić typ adresowanej pamięci, stosuje się odpowiednie rozkazy:

- MOV dla adresowania wewnętrznej pamięci RAM
- MOVX dla adresowania zewnętrznej pamięci RAM
- MOVC dla adresowania wewnętrznej i zewnętrznej pamięci programu (rozróżniane stanem wyprowadzenia EA oraz zakresem adresów).

Stany z obniżonym poborem mocy

Mikrokontrolery rodziny 51 mają mechanizmy pozwalające wprowadzić układ w stan, w którym nie jest wykonywany program, a pobór prądu zasilania jest ograniczony. W stanie jałowym (Idle) zegar taktujący zostaje odłączony, procesor jest zatrzymany (nie są wykonywane żadne rozkazy), normalnie pracują układ czasowo-licznikowy i port szeregowy oraz system przerwań, jeżeli był wcześniej odblokowany. Zawartość wszystkich rejestrów i pamięci wewnętrznej nie ulega zmianie. Pobór prądu zasilania jest 8 razy mniejszy niż w czasie normalnej pracy. Wyjście z tego stanu jest możliwe przez zerowanie systemu lub przerwanie zewnętrzne, jeżeli było wcześniej odblokowane. W stanie obniżonego poboru mocy Power Down następuje odłączenie zasilania od wszystkich bloków funkcjonalnych

z wyjątkiem pamięci wewnętrznej danych. Pobór prądu zasilania jest około 500 razy mniejszy niż w czasie normalnej pracy. Wyjście z tego trybu jest możliwe tylko na skutek wyzerowania mikrokontrolera.

4.3.2. Pytania sprawdzające

Odpowiadając na pytania, sprawdzisz, czy jesteś przygotowany do wykonania ćwiczenia.

1. Dlaczego mikroprocesory rodziny ’51 nazywamy mikrokontrolerami?
2. Jakie zasoby wewnętrzne posiadają mikrokontrolery rodziny ’51?
3. Jakie typy pamięci występują w mikrokontrolerze 8051. Do czego stosowany jest każdy z typów?
4. Czy w mikrokontrolerze 8051 jest zastosowana jednolita, czy rozdzielona przestrzeń adresowa?
5. Jakie obszary wyróżniamy w obrębie pamięci IRAM? W jaki sposób można adresować każdy z tych obszarów?
6. Ile portów równoległych posiada mikrokontroler 8051? Jakie jest ich przeznaczenie?
7. Jakie znaczenie mają poszczególne bity rejestru PSW?
8. Jaką rolę pełni każdy z sygnałów: PSEN, ALE, $\overline{EA}$, RST?
9. Które rozkazy zmieniają znaczniki w rejestrze PSW?
10. Gdzie znajduje się wynik operacji arytmetycznej?

4.3.3. Ćwiczenia

Ćwiczenie 1

Dobierz tryby adresacji do wybranych obszarów pamięci.

IRAM [0–7]	rejestrowa
IRAM [0–127]	rejestrowa pośrednia
XRAM [0–65535]	bezpośrednia
SFR	

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:

1) przeanalizować podane informacje,
2) dopasować metody adresacji do odpowiednich obszarów pamięci, ten sam tryb adresacji może się odnosić do więcej niż jednego obszaru,
3) zaprezentować wykonane ćwiczenie,
4) dokonać oceny poprawności wykonanego ćwiczenia.

Wyposażenie stanowiska pracy:

- treść zadania dla każdego ucznia,
- zeszyt przedmiotowy,
- literatura z rozdziału 6.

Ćwiczenie 2

Wypełnij tabelę, wpisując, jakie będą wartości rejestrów A i B (szesnastkowo) oraz flag po wykonaniu poleceń, jeżeli realizacja polecenia nie ma wpływu na zmianę wartości wpisz „–„:

Program	A	B	C	Z
załaduj do akumulatora liczbę 255 inkrementuj akumulator prześlij zawartość akumulatora do rejestru B				
wyzeruj flagę przeniesienia C załaduj do akumulatora liczbę 7 przesuń zawartość akumulatora o 1 bit w prawo prześlij zawartość akumulatora do rejestru B				
załaduj do akumulatora liczbę 0Fh prześlij zawartość akumulatora do rejestru B wykonaj iloczyn logiczny zawartości akumulatora z liczbą F0h				

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) przeanalizować każdy z zestawów poleceń,
2) obliczyć wartości, które zostaną umieszczone w rejestrach A i B po wykonaniu poleceń,
3) zastanowić się, które z poleceń mogą wpływać na zmiany flag C i Z, oraz ustalić jaką będą mieć wartość,
4) zaprezentować wykonane ćwiczenie,
5) dokonać oceny poprawności wykonanego ćwiczenia.

Wyposażenie stanowiska pracy:
- treść zadania dla każdego ucznia,
- literatura z rozdziału 6.

4.3.4. Sprawdzian postępów

Czy potrafisz:

		Tak	**Nie**
1)	omówić budowę mikrokontrolera 8051 na podstawie schematu blokowego?	☐	☐
2)	omówić zasoby wewnętrzne mikrokontrolera?	☐	☐
3)	wymienić wszystkie typy pamięci i omówić ich rolę?	☐	☐
4)	omówić sposób dołączenia pamięci zewnętrznych danych i programu do mikrokontrolera?	☐	☐
5)	scharakteryzować wybrane rejestry mikrokontrolera?	☐	☐
6)	podać znaczenie sygnałów sterujących w mikrokontrolerze?	☐	☐
7)	omówić sposoby obniżania poboru mocy mikrokontrolera i sposoby wychodzenia z tych stanów?	☐	☐

4.4. Programowanie mikrokontrolera w języku asemblera

4.4.1. Materiał nauczania

Lista rozkazów mikrokontrolerów rodziny ‘51

Oznaczenia:

R_r	–rejestry R0–R7,
direct	–adres w IRAM/ SFR,
R_i	–rejestry R0 lub R1 (używane w adresacji rejestrowej pośredniej),
Bit	–bity adresowalne w IRAM/ SFR,
data	–dana 8–bitowa,
data16	–dana 16–bitowa,
addr16	–adres w 64k,
addr11	–adres na stronie 2k,
Rel	–adres względny,
/bit	–negacja bitu.

Tabela 4 Operacje arytmetyczne i logiczne:

	mnemonik	Tryb adresacji				Flagi		
		rejestr.	bezpośr.	rej. pośr.	natychm.	C	AC	OV
operacje arytmetyczne								
Dodaj A←A+□	ADD	A,R_r	A,dir	A,@R_i	A,#data	x	x	x
Dodaj A←A+□+C	ADDC	A,R_r	A,dir	A,@R_i	A,#data	x	x	x
Odejmij A←A+□+C	SUBB	A,R_r	A,dir	A,@R_i	A,#data	x	x	x
Inkrementuj	INC	A/R_r/ DPTR	dir	@R_i				
Dekrementuj	DEC	A/R_r	dir	@R_i				
Mnóż A*B	MUL AB– wynik w B▪A					0		x
Dziel	DIV AB– wynik: A– wynik, B– reszta					0		x
Poprawka dziesiętna	DA A					x		
operacje logiczne								
Iloczyn log. AND	ANL	A,R_r dir,A	A,dir	A,@R_i	A,#data dir,#data			
Suma log. OR	ORL	A,R_r dir,A	A,dir	A,@R_i	A,#data dir,#data			
Modulo 2 XOR	XRL	A,R_r dir,A	A,dir	A,@R_i	A,#data dir,#data			
Neguj	CPL	A						
Zeruj	CLR	A						
Obrót w prawo	RR	A						
Obrót w lewo	RL	A						
Obrót w prawo z C	RRC	A				x		
Obrót w lewo z C	RLC	A				x		
Zamień 4 bity	SWAP	A						

Tabela 5 Operacje przesłań danych:

Kopiuj A←□	MOV	A,R_r	A,dir	A,@R_i	A,#data		
Kopiuj R_r←□	MOV	R_r,A	R_r,dir		R_r,#data		
Kopiuj dir←□	MOV	dir,A dir,R_r	dir,dir	dir,@R_i	dir,#data		
Kopiuj @R_i←□	MOV	@R_i,A	@R_i,dir		@R_i,#data		
Kopiuj DPTR←□	MOV				DPTR,#data16		
Kopiuj z pamięci programu	MOVC	A,@A+DPTR A,@A+DPTR					
Kopiuj z/do zewn. pam. danych	MOVX	A,@R_i A,@DPTR @R_i,A @DPTR,A					
Zamień	XCH	A,R_r	A,dir	A,@R_i			
Zamień mł. 4 bity	XCHD	A,@R_i					
Zapisz na stos	PUSH	dir SP=SP+1, (SP) ←dir					
Odczytaj ze stosu	POP	dir dir←(SP), SP=SP–1					

Tabela 6 Operacje na bitach:

Zeruj	CLR	C	bit
Ustaw	SETB	C	bit
Neguj	CPL	C	bit
Iloczyn logiczny	ANL	C,bit	C,/bit
Suma logiczna	ORL	C,bit	C,/bit
Kopiuj	MOV	C,bit	bit,C

Tabela 7 Skoki

Wywołanie podprogramu	LCALL addr16 ACALL addr11			
Powrót z podprogramu	RET			
Powrót z podprogramu obsługi przerwania	RETI			
Skok	LJMP addr16	AJMP addr11	SJMP rel	JMP @A+DPTR
Skok warunkowy	JC rel	JNC rel	JZ rel	JNZ rel
Skok warunkowy od bitu	JB bit,rel	JNB bit,rel	JBC bit,rel (po skoku bit=0)	
Porównaj, skocz jeśli ≠	CJNE A,dir,rel		CJNE A,#data,rel	
	CJNE R_r,#data,rel		CJNE @R_i,#data,rel	
Zmniejsz, skocz jeśli ≠ 0	DJNZ R_r,rel		DJNZ dir,rel	
Nic nie rób	NOP			

Każdy mikrokontroler działa zgodnie z programem umieszczonym w pamięci programu. Do napisania programu niezbędna jest znajomość języka niskiego poziomu – kodzie asemblera.
W przypadku mikrokontrolerów rodziny 51 można zastosować środowisko programowe m535 firmy Kail.

Etapy pisania i kompilowanie programu:

1. Przygotowanie algorytmu programu np. w postaci schematu blokowego.
2. Napisanie programu. Program należy napisać w edytorze tekstowym (np. notatniku) i zapisać pod nazwą składającą się z maksymalnie 8 znaków (bez spacji) z rozszerzeniem asm np. program.asm. Wielkość liter nie ma znaczenia zarówno w nazwie jak i w kodzie programu.

3. Linia programu ma postać:
   [etykieta:] mnemonik rozkazu [operand],[operand] [;komentarz]
   Liczby przedstawiamy dziesiętnie, binarnie – wówczas liczbę należy zakończyć literą b lub szesnastkowo – liczbę należy zakończyć literą h. Argument szesnastkowy rozpoczynający się od litery musi być poprzedzony cyfrą 0.
4. Program powinien rozpoczynać się od dyrektywy org (np. org 0– kompilator umieści program od adresu 0), a kończyć dyrektywą end (do tego miejsca program będzie kompilowany).
5. Po napisaniu programu należy go skompilować a51.exe **nazwa_programu.asm**. Asembler sprawdza, czy zbiór źródłowy nie ma błędów w składni, za nazwy stałych podstawia odpowiednie wartości liczbowe, za etykiety – adresy. Następnie generuje dwa zbiory wynikowe:
   nazwa_programu.lst, w którym zbiór źródłowy jest rozszerzony o adresy i kody rozkazów, numerację wierszy, tablicę symboli wraz z wartością lub adresem pod którym występują, wskazane są również ewentualne błędy,
   nazwa_programu.obj, który jest plikiem wejściowym dla linkera lub konwertera INTEL–HEX.
6. Jeżeli program wynikowy ma być w standardzie INTEL–HEX należy wywołać konwerter ohs51.exe nazwa_programu.obj, wynikiem będzie plik nazwa_programu.hex.
7. Program może być podzielony na segmenty związane z obszarem pamięci, w którym mają być umieszczone. Segmenty mogą być absolutne (czyli takie, których adresy są ustalone przez programistę) lub relokowalne (czyli takie, których adresy są ustalane na etapie łączenia modułow programu)
8. Jeżeli program wynikowy ma być połączony z wielu modułów należy zastosować linker, który połączy relokowalne segmenty w jeden segment, ustali adresy absolutne i utworzy zbiór wyjściowy zawierający cały program. Wywołanie linkera odbywa się przez podanie jego nazwy (l51.exe), listy nazw zbiorów wejściowych w kolejności, w jakiej mają być łączone moduły i opcjonalnie nazwy zbioru wyjściowego. Jeżeli nie podamy nazwy zbioru wyjściowego to zostanie nadana nazwa pierwszego wejściowego:
   l51.exe zb1.obj, zb2,obj,….[to zb_wy.abs]
9.

Dyrektywy asemblera:

1. Dyrektywy definiujące symbole
   Dyrektywa **SEGMENT**

Dyrektywą tą można zadeklarować nazwy symboliczne segmentów relokowalnych i ich typ (tzn. w obszarze jakiego typu pamięci ma być umieszczony) oraz ewentualnie sposób relokacji. Nazwa segmentu jest nazwą symboliczną – oznacza adres, od którego rozpoczyna się ten segment

Postać ogólna dyrektywy:

nazwa _ segmentu SEGMENT typ _ segmentu [sposób _ relokacji]

np: PROG_1 SEGMENT CODE

Typ segmentu określa obszar pamięci, gdzie deklarowany segment ma być umieszczony:

CODE – obszar pamięci programu (PGM),
XDATA – obszar zewnętrznej pamięci danych,
DATA – obszar wewnętrznej pamięci danych adresowany bezpośrednio (adres 0÷127),
IDATA – obszar wewnętrznej pamięci danych adresowanych pośrednio (adres 0÷127),
BIT – obszar wewnętrznej pamięci danych adresowany bitowo (32 47– adresy bitów 0 127).

Sposób relokacji – dotyczy sposobu zapisu pliku po linkowaniu w pamięci CODE. Bez podania sposobu relokacji pliki są umieszczane w pamięci kolejno, bez odstępów, w pamięci CODE. Podając sposób relokacji można wprowadzać odstępy. Możliwości relokacji:
INPAGE – na stronie
INBLOCK – w bloku,
BITADDRESSABLE – w obszarze pamięci adresowanej bitowo,
UNIT – bez odstępów – domyślnie, jeżeli nie podany inny sposób relokacji.

Dyrektywa EQU

Dyrektywa ta pozwala na nadawanie symbolowi (nazwie symbolicznej) wartości. Definiuje ona tzw. stałe kompilacji tzn. wartość danego symbolu nadana dyrektywą EQU nie może zostać zmieniona ani taką samą dyrektywą, ani żadną inną w całym programie.
Ogólna postać: nazwa_symbolu EQU wartość_symbolu
Nadawanie symbolowi wartości może być dokonywane jednym z trzech sposobów:

podanie konkretnej wartości np:

```
LIMIT EQU 1200
R8 EQU 08
```

podanie symbolu wcześniej znanego np:

```
KONIEC EQU LIMIT+5
WARTOSC EQU KONIEC+'A'
```

podanie symbolu znanego asemblerowi (np. symbolicznej nazwy rejestru z SFR) np:

```
SERIAL EQU SBUF
LICZNIK EQU R5
BUSY EQU ACC.7
```

Dyrektywa SET

Dyrektywa ta nadaje wartości symbolowi (nazwie symbolicznej). Pozwala ona na zdefiniowanie tzw. zmiennych kompilacji tzn. takich symboli, których wartości można zmieniać w programie inną dyrektywą SET powołując się na nazwę (symbolu).
Ogólna postać: nazwa_ symbolu SET wartość_symbolu
Nadanie wartości symbolowi jest możliwe na 3 różne sposoby:

przez podanie konkretnej wartości np:

```
VALUE SET 100
```

przez podanie symbolu wcześniej znanego np:

```
LIMIT EQU 1200
VALUE SET LIMIT–200
```

przez podanie symbolu znanego asemblerowi np:

```
COUNTER SET R1
```

Dyrektywa SET odnosząca się do jednej nazwy symbolicznej może wystąpić wielokrotnie w danym programie. Dyrektywy SET i EQU definiują nazwy symboliczne w tym obszarze pamięci, na który jest nastawiony wskaźnik w momencie ich występowania w programie.

Dyrektywa BIT

Dyrektywa BIT pozwala na zdefiniowanie nazwy symbolicznej bitu i podanie jego adresu tzn. przyporządkowuje nazwie bitu adres z przestrzeni pamięci danych adresowanej bitowo.
Postać ogólna: Nazwa_bitu BIT adres_bitu
Adres bitu może być podany w następujący sposób:

jako adres absolutny bitu, przy czym może on być podany jako adres kolejnego bitu w przestrzeni adresowej lub jako numer bitu w komórce pamięci adresowanej bitowo. Komórka ta musi leżeć w obszarze wewnętrznej pamięci danych adresowanym bitowo tzn. w obszarze $32 \div 47$ ($20_H \div 2F_H$) np.:

```
X_OFF   BIT  24H.2
```

jako adres w postaci symbolicznej zdefiniowany wcześniej poprzez symbol (tzn. nazwę symboliczną bitu) np.:.

```
ALARM   BIT   X_OFF+1
```

jako adres w postaci symbolicznej poprzez symbol znany asemblerowi np.:

```
KLAWISZ   BIT  P1.0
ZNACZNIK   BIT  C
BSY BIT ACC.7
```

Dyrektywy DATA (IDATA, XDATA, CODE).

Są to dyrektywy definiujące wartość symbolu (nazwy symbolicznej) w obszarach pamięci danych i programu:

DATA – wewnętrzna pamięć danych,
IDATA – wewnętrzna pamięć danych,
XDATA – zewnętrzna pamięć danych,
CODE – pamięć programu.

Postać ogólna: Nazwa symbolu DATA wartość

Wartość może być podana jako :

wartość bezwzględna np:

```
RESULT  DATA  40H
```

symbol (nazwa symboliczna) zdefiniowana poprzednio np:

```
WYNIK  DATA  RESULT+1
```

symbol znany asemblerowi (z zasobów programowych) np:

```
REJ_SZER  DATA  SBUF
```

Analogicznie dla pozostałych obszarów pamięci np.:

```
BUFFER  IDATA  60H
TIME  XDATA  100H
RESTART  CODE  00H
INT_VEC_1  CODE  RESTART+0BH
```

Dyrektywy DATA, IDATA, XDATA, CODE pozwalają na nadanie symbolowi wartości oraz atrybutu umieszczenia go w odpowiedniej przestrzeni adresowej (tzn. w odpowiednim typie pamięci).

Dyrektywy rezerwacji i inicjacji pamięci (w aktywnym segmencie).

Dyrektywa DS – define storage, dyrektywa ta pozwala na rezerwację określonej ilości bajtów pamięci

Ogólna postać: [etykieta:] DS ilość_bajtów

np:

```
GAP:  DS  20
      DS  5
```

Dyrektywa DB – define bajt

Dyrektywa DB inicjuje (bajtowo) poszczególne komórki pamięci tzn. pozwala zapisać w nich liczby lub znaki.

Ogólna postać: [etykieta] DB wartość [, wartość1, ... ,] np.:

```
LAB: DB  'WCIŚNIJ  DOWOLNY  KLAWISZ',0
```

REG: DB 0, 1, 8, ‘A’, ‘0’, ‘; ‘

Dyrektywa DW – define word.

Definicja pozwala inicjować 2 kolejne komórki pamięci np.:

TAB: DW 5AF3H

LICZBY: DW 1, 2, 3 – zapis każdej liczby w pamięci na 2 bajtach będzie miał postać 0001H, 0002H itd.

Dyrektywa ta często stosowana jest do definiowania adresów programów.

Dyrektywa DBIT pozwalająca na określenie wartości bitu.

Postać ogólna: [etykieta:] DBIT wartość

Wartość bitu można określić 3 sposobami, a mianowicie:

wartość konkretna (domyślnie tylko 0 lub 1) np:

```
KOD: DBIT  0, 1, 1, 0,
```

poprzez nazwę symboliczną wcześniej zdefiniowaną np:

```
PAL   EQU 1b
TEN_BIT   DBIT   PAL
DBIT   NOT ( PAL )
```

nazwę znaną asemblerowi np:

```
INNY_BIT   DBIT   P0.0
```

Dyrektywy udostępniające nazwy

Dyrektywy udostępniające nazwy to: **PUBLIC**, **EXTERN**, **NAME** . wykorzystywane one są w programach wielomodułowych.

Dyrektywa PUBLIC (upubliczniona)

Służy do przekazania nazwy symbolicznej do innego modułu, aby nazwa ta była widoczna na zewnątrz i można było wykorzystać jej wartość w innym module.

Postać ogólna: PUBLIC symbol [, symbol] np :

```
PUBLIC   PRINT_STR
```

Dyrektywa EXTERN

Dyrektywa ma zastosowanie, gdy używamy w pliku symboli zdefiniowanych w innych plikach (upublicznionych dyrektywą PUBLIC). Używając symbolu z zewnątrz należy zapewnić tzw. zgodność atrybutów to znaczy, że zapis danego symbolu jest w tym samym obszarze pamięci (taki sam typ), w którym będzie szukana.

Postać ogólna: EKSTERN Typ_segmentu (lista_symbol) np:

```
EKSTERN   CODE   (A2B, B2A )
EKSTERN   DATA   ( FLAGI )
```

Dyrektywa NAME

Dyrektywa NAME służy do deklaracji nazwy modułu.

Postać ogólna: NAME nazwa_obiektu

Nazwa obiektu może posiadać do 40 znaków.

Dyrektywy sterujące.

Dyrektywami sterującymi są: **END**, **USING**, **ORG**, **RSEG**, **CSEG**, **DSEG**, **XSEG**, **ISEG**, **BESEG**.

Dyrektywa **END**

Dyrektywa ta określa koniec modułu programu. Musi nią być zakończony moduł. Po identyfikacji dyrektywy END w pliku źródłowym kompilator kończy tłumaczenie programu.

Dyrektywa **USING**

Służy ona do wyboru bieżącego rejestru. Adresy tych rejestrów są określone przez nazwy symboliczne AR0 ÷ AR7. Dzięki temu można jednocześnie wykorzystywać 2 banki rejestru (jeden wynikający z USING a drugi z PSW (RS1,RS0).

Dyrektywa **ORG**

Nadaje ona wartość wskaźnika w bieżąco rozpatrywanym segmencie.

Postać ogólna: ORG argument

Argument ma tu znaczenie adresu pamięci (bajtowo) np.:

ORG 100H

moduł zostanie skompilowany i umieszczony w pamięci programu od adresu 100H.

Dyrektywa RSEG

Kompilator umieści wszystkie segmenty relokowalne o danym typie w tym samym typie pamięci.

Postać ogólna: RSEG nazwa_symbol_segment_relokowalny np.:

DATA_SEG SEGMENT DATA – określenie segmentu w pamięci DATA

RSEG DATA_SEG

CODE_SEG SEGMENT CODE – określenie segmentu w pamięci DATA

RSEG CODE_SEG

Dyrektywy ustalające absolutny segment

CSEG – segment pamięci programu jako bieżący,

DSEG – segment pamięci danych adresowanych bajtów,

XSEG – segment pamięci zewnętrznej danych,

ISEG – segment pamięci danych adresowanych pośrednio,

BSEG – segment pamięci danych dostępnych bitowo.

Makrodefinicje

Makro to zestaw instrukcji asemblera. Ciąg instrukcji występujący po linii zawierającej dyrektywę MACRO, aż do najbliższej dyrektywy ENDM, tworzy makro o określonej nazwie. Po zdefiniowaniu cały taki zestaw może być włączony w kod źródłowy programu poprzez wywołanie makra. W treści makr mogą występować bez ograniczeń wywołania innych makr, ale nie może wystąpić definicja innego makra. Makro jest wywoływane poprzez umieszczenie jego nazwy w polu rozkazu danej linii programu. Przy wywołaniu podawane są parametry aktualnego makra. W czasie wstawiania makra w kod programu asembler zastępuje wszystkie parametry formalne parametrami aktualnymi.

Postać ogólna: Nazwa_makro MACRO [parametry]

Przykład:

```
CLEAR  MACRO  G1, G2  –  makrodefinicja – zapis pod adres G1 wartości G2
MOV  R0, # G1                  |
MOV  A, # G2                   |
MOV  @ R0, A                   |– tzw. ciało makrodefinicji
ENDM                           |
```

Wywołanie:

```
CLEAR  20H,0
CLEAR   5,'A'
```

Instrukcje sterujące języka asembler 51

Są to instrukcje umieszczone w pliku źródłowym programu, służące do określenia dodatkowych warunków tłumaczenia (kompilacji) programu. Instrukcje te są poprzedzone znakiem $.

$ INCLUDE – pozwala na włączenie do tłumaczonego pliku innego pliku źródłowego np:

$ INCLUDE (REG.535.INC)

$ DEBUG / NODEBUG ($DB/ NDB)– dodawanie do pliku objaśnień wszystkich nazw symbolicznych (lokalnych i publicznych)

$ ERRORPRINT/NOERRORPRINT ($EP, $NEP) wskazanie nazwy pliku, w którym mają być umieszczane błędy i opisy błędów np:

$ EP (BLEDY.ERR)

4.4.2. Pytania sprawdzające

Odpowiadając na pytania, sprawdzisz, czy jesteś przygotowany do wykonania ćwiczeń.

1. W jaki sposób zmieniają się zawartości rejestrów, komórek pamięci i znaczniki na skutek wykonania poszczególnych rozkazów (przeanalizuj na kilku przykładach)?
2. Jaki będzie wynik działania analizowanego fragmentu programu?
3. Jakie są zasady pisania programu w języku asemblera?
4. W jaki sposób skompilować program i poprawić ewentualne błędy?
5. W jaki sposób uruchomić program używając zestawu uruchomieniowego korzystając z pracy krokowej, ciągłej i pułapek?
6. Jak wykorzystywać w programach istniejące podprogramy?
7. Do czego służą dyrektywy asemblera, na przykład ORG, DB, END, EQU, SET?

4.3.3. Ćwiczenia

Ćwiczenie 1

Napisz i uruchom programy przemieszczania tablic pomiędzy różnymi typami pamięci:

- W pamięci IRAM od adresu 20h znajduje się tablica 30 danych. Napisz i uruchom program, który przepisze tablicę do pamięci XRAM od adresu 0.
- W pamięci programu znajduje się tablica zakończona znacznikiem końca – liczbą FFH. Napisz i uruchom program, który przepisze tablicę do pamięci IRAM od adresu 20H.
- Napisz i uruchom program, który przepisze z tablicy w pamięci XRAM od adresu 0 do IRAM od adresu 20.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:

1) opracować algorytm każdego programu i przedstawić go w postaci schematu blokowego,
2) napisać programy w języku asemblera,
3) skompilować programy, poprawić ewentualne błędy,
4) uruchomić programy używając zestawu uruchomieniowego,
5) sprawdzić poprawność działania programów,
6) zaprezentować wykonane ćwiczenie,
7) dokonać oceny poprawności wykonanego ćwiczenia.

Wyposażenie stanowiska pracy:
- komputer PC,
- oprogramowanie umożliwiające kompilowanie i debbugowanie programu,
- zestaw uruchomieniowy z oprogramowaniem,
- literatura z rozdziału 6.

Ćwiczenie 2

W pamięci IRAM znajdują się dwie tablice danych: TAB1 od adresu 20h i TAB2 od adresu 30h. Obie tablice zawierają po 10 danych jednobajtowych liczb dziesiętnych. Napisz program, który obliczy iloczyny par liczb pobranych z obu tablic, a wyniki w postaci dwubajtowych liczb dziesiętnych zapisze do pamięci XRAM od adresu 0. Wyniki zapisać w kolejności: starszy bajt – młodszy bajt.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) opracować algorytm programu i przedstawić go w postaci schematu blokowego,
2) w celu konwersji liczb z postaci dziesiętnej na szesnastkową i konwersji wyniku z szesnastkowego na dziesiętny wykorzystać gotowe programy konwersji lub napisać własne podprogramy,
3) napisać program w języku asemblera,
4) skompilować program, poprawić ewentualne błędy,
5) uruchomić program używając zestawu uruchomieniowego,
6) sprawdzić poprawność działania programu,
7) zaprezentować wykonane ćwiczenie,
8) dokonać oceny poprawności wykonanego ćwiczenia.

Wyposażenie stanowiska pracy:
- komputer PC,
- oprogramowanie umożliwiające kompilowanie i debbugowanie programu,
- zestaw uruchomieniowy z oprogramowaniem,
- literatura z rozdziału 6.

Ćwiczenie 3

W parach rejestrów R4–R5 i R6–R7 znajdują się liczby dwubajtowe. Napisz program, który, w zależności od zawartości rejestru R4, wykona odpowiednie działanie na liczbach dwubajtowych i wynik działania umieści w pamięci XRAM od adresu 0.

Działania:

R4=0 – dodawanie,

R4=1 – odejmowanie,

R4=2 – mnożenie,

R4=3 – dzielenie (część całkowita i reszta z dzielenia).

* wykonaj zadanie dla liczb w zapisie znak– moduł (pierwszy bajt– bajt znaku 0– dodatni, 1– ujemny, bajty drugi i trzeci– wartość bezwzględna liczby), wyniki zapisz w postaci liczb pięciobajtowych (1 bajt–znak, cztery następne– wartość bezwzględna liczby).

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) opracować algorytm programu i przedstawić go w postaci schematu blokowego,
2) napisać program w języku asemblera,

3) skompilować program, poprawić ewentualne błędy,
4) uruchomić program używając zestawu uruchomieniowego,
5) sprawdzić poprawność działania programu,
6) zaprezentować wykonane ćwiczenie,
7) dokonać oceny poprawności wykonanego ćwiczenia.

Wyposażenie stanowiska pracy:
- komputer PC,
- oprogramowanie umożliwiające kompilowanie i debbugowanie programu,
- zestaw uruchomieniowy z oprogramowaniem,
- literatura z rozdziału 6.

Ćwiczenie 4

Przy pomocy dyrektyw asemblera zadeklaruj 16 bajtów w pamięci IRAM jako obszar stosu, a następnie ustal wskaźnik stosu i wypróbuj działanie w programie zawierającym wywołanie podprogramu i instrukcje PUSH i POP.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zadeklarować stos jako segment w pamięci IRAM,
2) określić jego rozmiar – zarezerwować 16 bajtów,
3) ustalić wskaźnik stosu rozkazem MOV SP,#…. pamiętając, że przed złożeniem danej na stos wskaźnik jest inkrementowany,
4) napisać prosty program zawierający co najmniej jedno wywołanie podprogramu i parę instrukcji PUSH i POP,
5) skompilować program, poprawić ewentualne błędy,
6) uruchomić program używając zestawu uruchomieniowego,
7) sprawdzić poprawność działania programu,
8) zaprezentować wykonane ćwiczenie,
9) dokonać oceny poprawności wykonanego ćwiczenia.

4.3.4. Sprawdzian postępów

Czy potrafisz:

		Tak	**Nie**
1)	omówić działanie poszczególnych rozkazów z listy rozkazów?	☐	☐
2)	dobrać odpowiednie rozkazy do zaplanowanego działania?	☐	☐
3)	napisać program w języku asemblera?	☐	☐
4)	skompilować program i poprawić błędy?	☐	☐
5)	uruchomić program używając zestawu uruchomieniowego, poprawić jego działanie?	☐	☐
6)	stosować dyrektywy asemblera w pisanych programach?	☐	☐
7)	używać podprogramów, definiować makrodefinicje?	☐	☐
8)	stosować linker do łączenia segmentów programu?	☐	☐

6. LITERATURA

1. Dyrcz K., Kowalski C. T., Zarczyński Z.: Podstawy techniki mikroprocesorowej, Oficyna Wydawnicza Politechniki Wrocławskiej, 1999.
2. Janiczek J., Stępień A.: Mikrokontroler 80(C)51/52, Wydawnictwo Elektronicznych Zakładów Naukowych, Wrocław 1995.
3. Janiczek J., Stępień A.: Laboratorium systemów mikroprocesorowych cz. I i II, Wydawnictwo Elektronicznych Zakładów Naukowych, Wrocław 1995.
4. Lipowski J., Małysiak H., Pochopień B., Podsiadło P., Wróbel E.: Modułowe systemy mikrokomputerowe, Wydawnictwa Naukowo– Techniczne, Warszawa 1984.
5. Rydzewski A.: Mikrokomputery jednoukładowe rodziny MCS–51,Wydawnictwa Naukowo– Techniczne, Warszawa 1992.
6. Starecki T.: Mikrokontrolery 8051 w praktyce, Wydawnictwo BTC, Warszawa 2000
7. http://zsk.tech.us.edu.pl.