4. MATERIAŁ NAUCZANIA

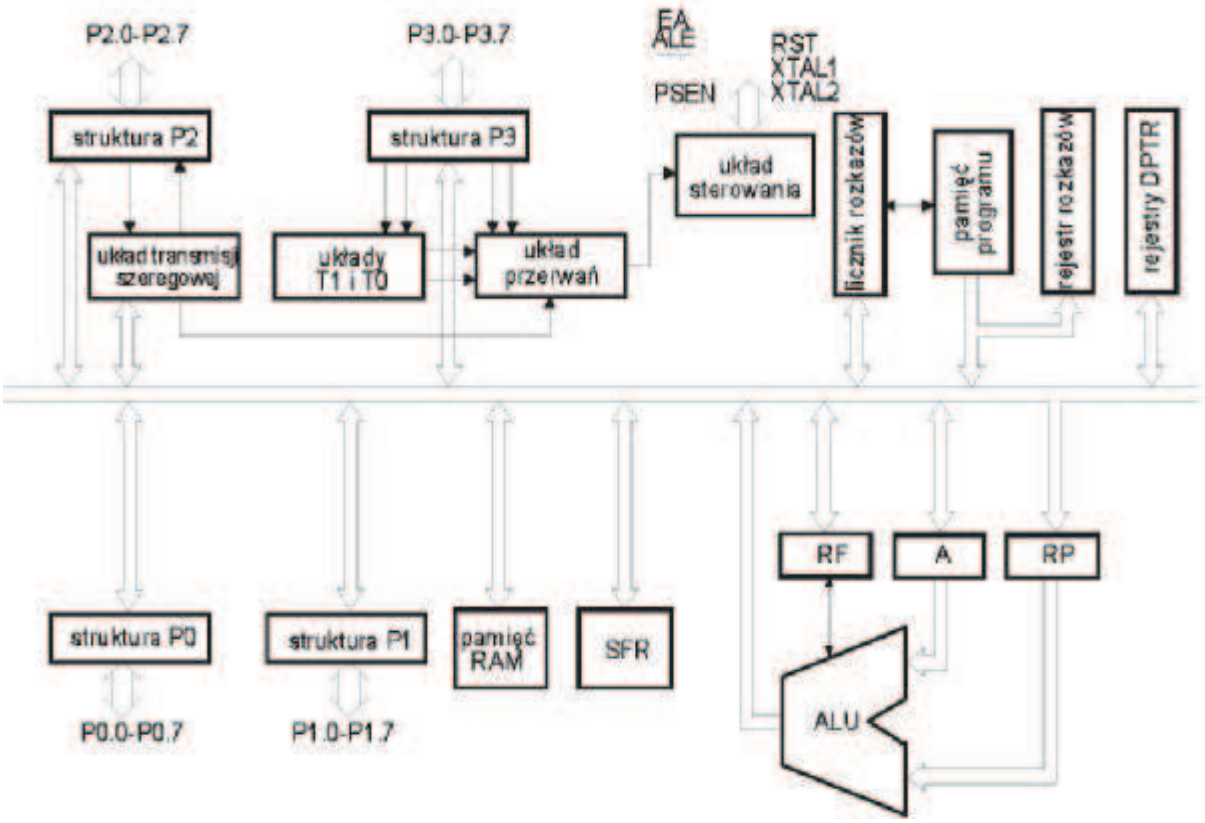
4.1. Mikrokontroler 8051

4.1.1. Materiał nauczania

Mikrokontroler - to komputer wykonany w jednym układzie scalonym, używany do sterowania urządzeniami elektronicznymi. Układ oprócz jednostki centralnej CPU posiada zintegrowaną różnego rodzaju pamięć oraz układy wejścia wyjścia. Mikrokontroler stanowi użyteczny i całkowicie autonomiczny system mikroprocesorowy, który z reguły nie potrzebuje współpracować z układami zewnętrznymi.

Mikrokontroler 8051

Jednym z najbardziej rozpowszechnionych 8-bitowych systemów mikroprocesorowych jest jednoukładowy komputer 8051 (firmy Intel) zwany również mikrokontrolerem 8 bitowym, którego schemat blokowy przedstawiono na rys.1



Rys.1. Uproszczony schemat blokowy mikrokontrolera 8051

Elementy architektury układu 8051

W mikrokontrolerze 8051 możemy wyróżnić następujące bloki składowe:

- układ arytmetyczno-logiczny (ALU, RF, A, RP)
- blok rejestrów specjalnych (SFR),
- pamięć danych (RAM) 128B,
- pamięć programu (EROM) 4kB,
- porty wejścia/wyjścia (P0, P1, P2 i P3),

- programowany układ czasowy (T0, T1),
- układ transmisji szeregowej,
- blok przerwań,
- generator sygnału taktującego,
- układ sterowania.

Układ arytmetyczno - logiczny

Podstawowymi elementami składowymi układu arytmetyczno-logicznego są:

- 8-bitowa jednostka przetwarzająca (ALU) z układem korekcji dziesiętnej,
- rejestry pomocnicze używane przy wykonywaniu obliczeń (niedostępne dla programisty),
- dekoder rozkazów.

Z układem arytmometru współpracują ponadto dwa rejestry operacyjne: akumulator (A) i rejestr pomocniczy RP oznaczany również B. Akumulator zawiera jeden z operandów oraz zapisywany jest w nim wynik operacji. Rejestr RP zawiera drugi operand wykorzystywany głównie przy operacjach mnożenia i dzielenia. Argumentami operacji wykonywanych przez arytmometr mogą być również zawartości innych rejestrów specjalnych, zawartości komórek pamięci lub dane wprowadzane bezpośrednio.

Arytmometr (ALU) może wykonywać następujące operacje na argumentach ośmiobitowych:

- dodawanie,
- dodawanie z przeniesieniem,
- odejmowanie z pożyczką,
- zwiększanie i zmniejszanie zawartości akumulatora,
- mnożenie w naturalnym kodzie binarnym dające 16-bitowy wynik,
- dzielenie w naturalnym kodzie binarnym dające 8-bitowy wynik i 8-bitową resztę,
- iloczyn logiczny, suma logiczna i suma modulo 2,
- zerowanie, negacja i rotacja zawartości akumulatora,
- korekcja dziesiętna zawartości akumulatora.

Z arytmometrem ściśle powiązany jest rejestr flagowy. Wartości bitów w rejestrze flagowym zmieniają się w czasie wykonania przez arytmometr operacji arytmetyczno-logicznej i opisują rezultaty ostatnio wykonywanego rozkazu. Miejsca występowania poszczególnych bitów rejestru flagowego przedstawiono na rys. 2. Większość bitów flagowych jest identyczna z flagami innych procesorów 8-bitowych:

- P (parity) - to znacznik parzystości wartości bitów zapisanych akumulatorze, 1 – oznacza parzysta liczbę jedynek w zapisanych w akumulatorze, 0 – nieparzystą,
- OV (overflow) - znacznik przepełnienia (nadmiaru) dla dodawania i odejmowania w kodzie U2, 1 – oznacza wystąpienie nadmiaru bądź przepełnienia, 0 – jego brak,
- RSO, RS1 - to bity wyboru banku rejestrów roboczych, według kodu podanego na rys.3,
- F0 - jest znacznikiem uniwersalnym dowolnego zastosowania,
- AC (auxiliary carry) - znacznik przeniesienia połówkowego; ustawiany w przypadku wystąpienia przeniesienia miedzy bitami 3 i 4 akumulatora, 1 – wystąpienie przeniesienia, 0 – jego brak,
- C (CY) (carry) - jest znacznikiem przeniesienia ustawianym w przypadku przepełnienia siódmego bitu akumulatora b_7, 1 – wystąpienie przeniesienia, 0 – jego brak.

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
C	AC	F0	RS1	RS0	OV	-	P

Rys.2. Rejestr flagowy mikrokontrolera 8051 [7]

RS1	RS0	Wybrany bank
0	**0**	bank 0
0	**1**	bank 1
1	**0**	bank 2
1	**1**	bank 3

Rys.3. Znaczenie bitów RS0 i RS1 [7]

Pamięć danych

Wewnętrzna pamięć danych zawiera dwa bloki:

- 128 bajtów ciągłego obszaru pamięci RAM (rys. 4),
- 128 bajtów bloku rejestrów specjalnych (SFR) mikrokontrolera.

Obszar od adresu 0 do 31 (0H-1FH) zajmują cztery banki rejestrów roboczych, po osiem rejestrów w banku (adresowane bitami RS1 i RS0 rejestru flagowego). Rejestry oznaczone symbolami R0 - R7 mogą być wykorzystywane do przechowywania danych. Rejestry R0 i R1 każdego bloku, mogą być wykorzystane do indeksowego adresowania wewnętrznej i zewnętrznej pamięci danych. Po wyzerowaniu mikrokontrolera sygnałem RST użytkownik dysponuje blokiem 0 rejestrów roboczych. Jest to obszar pamięci o adresach 32-47(20H-2FH), który może być używany do przechowywania dowolnych danych wykorzystywanych w programie. Cechą tego obszaru jest możliwość zaadresowania dowolnego, pojedynczego bitu komórki pamięci, w obszarze 0-127 (0H-7FH), natomiast numer bitu wskazujemy z wyliczenia: (n-32)*8+i (n jest adresem bajtu, *i* numerem bitu w tym słowie). Mapę pamięci wewnętrznej RAM przedstawiono na rys.4.

Adresy 48-127 (30H-7FH)	Pamięć danych użytkownika
Adresy 32-47 (20H-2FH)	Pamięć adresowana bitowo (adresy bitów 0-127)
Adresy 24-31 (18H-1FH)	Blok 3 - rejestry R0-R7
Adresy 16-23 (10H-17H)	Blok 2 - rejestry R0-R7
Adresy 8-15 (8H-0FH)	Blok 1 - rejestry R0-R7
Adresy 0-7 (0H-7H)	Blok 0 - rejestry R0-R7

Rys.4. Mapa pamięci RAM mikrokontrolera 8051 [7]

Do adresacji pojedynczych bitów odwołują się specyficzne rozkazy mikrokontrolera.

W wewnętrznej pamięci danych umieszczany jest również stos, lokowany w dowolnym miejscu pamięci wewnętrznej sterownika. Po wyzerowaniu układu sygnałem RST zawartość wskaźnika stosu *(Stack Pointer)* SP = 7. Wskaźnik stosu jest inkrementowany przed każdym zapisem na stos i dekrementowany po każdym odczycie.

Oprócz bloku pamięci wewnętrznej mikrokontroler może współpracować z 64 KB zewnętrznej pamięci. Ponieważ nie dysponuje on niezależnymi rozkazami komunikacji z urządzeniami WE/WY, w obszarze adresowym 64KB pamięci zewnętrznej muszą być również uwzględnione rejestry urządzeń WE/WY.

Blok rejestrów specjalnych mikrokontrolera 8051

Blok rejestrów specjalnych (**SFR** - Special Function Registers) umieszczono w tzw. niespójnym obszarze pamięci sterownika, o adresach 128-240 (80H-0F0H). Są one wykorzystywane na dwa sposoby: są tam lokowane wszystkie (za wyjątkiem licznika rozkazów i czterech banków rejestrów R0-R7) rejestry sterujące pracą układu lub wykorzystywane są bezpośrednio do zapisu programu. Rejestry SFR mogą również pracować jako interfejs pomiędzy mikroprocesorem a układami zewnętrznymi sterownika. Dostęp do każdego z tych rejestrów odbywa się w trybie adresowania bezpośredniego. Wykaz rejestrów specjalnych sterownika zamieszczono w tabl. 1.

Tabela 1 Rejestry specjalne (SFR) mikrokontrolera 8051 [7]

Nazwa	Adres	Pełniona funkcja
PO	128 (80H)	Port WE/WY P0
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 procesora
TCON	136 (88H)	Rejestr sterujący układów czasowych T0 i T1
TMOD	137 (89H)	Rejestr trybu pracy układów czasowych T0 i T1
TLO	138 (8AH)	Rejestr danych układu czasowego T0 (mniej znaczący)
TL1	139 (8BH)	Rejestr danych układu czasowego T1 (mniej znaczący)
THO	140 (8CH)	Rejestr danych układu czasowego T0 (bardziej znaczący)
TH1	141 (8DH)	Rejestr danych układu czasowego T1 (bardziej znaczący)
P1	144 (90H)	Port WE/WY T1
SCON	152 (98H)	Rejestr sterujący układu transmisji szeregowej
SBUF	153 (99H)	Rejestr danych układu transmisji szeregowej
P2	160(0A0H)	Port WE/WY T2
IE	168(0A8H)	Rejestr maski przerwań
P3	176(0B0H)	Port WE/WY T3
IP	184(0B8H)	Rejestr priorytetów przerwań
PSW	208 (0D0H)	Słowo stanu procesora/Flagi
A	224 (OEOH)	Akumulator
RP	240 (OFOH)	Rejestr ogólnego przeznaczenia
Rejestry specjalne mikrokontrolera 8051.		

W standardowym układzie 8051 zainstalowano 4KB programowalnej pamięci typu EPROM (niektóre wersje układu wyposażono w pamięć EEPROM). W pamięci stałej zapisywane są kody stałych operacji kontrolera. Pamięć programu adresowana jest 16-bitowym licznikiem rozkazów (PC). Mikrokontroler 8051 może także korzystać z 64KB pamięci zewnętrznej. Wyborem rodzaju pamięci steruje linia $\overline{EA}$.W przypadku korzystania z wewnętrznej pamięci programu sygnał $\overline{EA}$ musi mieć wartość wysoką. Jeśli mikrokontroler ma korzystać z zewnętrznej pamięci programu linia $\overline{EA}$ musi być ustawione w stan niski.

Ponieważ zerowanie sygnałem RST ustawienie licznik rozkazów w stan 0000H, początek programu musi być umieszczony pod tym samym adresem. Zwykle umieszczana jest tam instrukcja skoku do dalszego obszaru pamięci programu, ponieważ począwszy od adresu 0003H pierwsze kilkadziesiąt bajtów (Tabl. 2) rezerwowane jest przez procedury obsługi przerwań.

Tabela 2 Adresy wektorów przerwań sterownika 8051 [7]

Adres	Zawartość
0003H	Początek procedury obsługi przerwania zewnętrznego
000BH	Początek procedury obsługi przerwania z układu
0013H	Początek procedury obsługi przerwania zewnętrznego
001BH	Początek procedury obsługi przerwania z układu
0023H	Początek procedury obsługi przerwania z układu transmisji

Licznik PC zawiera adres aktualnego rozkazu przeznaczonego do wykonania. Kod rozkazu przekazywany do rejestru rozkazów (jednostki sterującej) steruje wyborem źródła

argumentów, miejsca umieszczenia wyniku, funkcjami arytmometru, itp. Jeżeli nie jest wykonywany rozkaz skoku, to zawartość licznika rozkazów jest inkrementowana po odczycie każdego bajtu z pamięci programu

Porty wejścia/wyjścia

Są to linie we/wy mikrokontrolera 8051 pogrupowane są w cztery 8-bitowe porty:

1) P0 - linie P0.0/AD0 - P0.7/AD7
2) P1 - linie P1.0 - P1.7
3) P2 - linie P2.0/A8 - P2.7/A15
4) P3 - linie P3.0/RxD', P3.1/TxD', P3.2/INT0', P3.3/INT1', P3.4/T0, P3.5/T1, P3.6/WR', P3.7/RD'

Wszystkie linie portów P0-P3 pracujące jako standardowe linie wejścia/wyjścia są niezależne pod względem kierunku przesyłania informacji. Rejestry P0-P3 złożone z przerzutników poszczególnych linii wchodzą w skład bloku rejestrów specjalnych, przy czym możliwe jest adresowanie ich poszczególnych bitów, co umożliwia bezpośrednie sterowanie pojedynczymi liniami we/wy,

Programowany układ czasowy

Mikrokontroler 8051 wyposażony jest w dwa układy licznikowe T0 i T1.Każdy z tych liczników składa się z dwóch ośmiobitowych połówek. Połówki te są widziane przez mikroprocesor jako rejestry specjalne TH0 i TL0 dla układu T0 oraz TH1 i TL1 dla układu T1. Każdy z obu liczników może pracować jako licznik (zlicza wówczas impulsy zewnętrzne) lub jako czasomierz (zlicza cykle maszynowe mikrokontrolera). Układy licznikowe mogą pracować w czterech trybach. Wybór trybu pracy i sterowanie zliczaniem odbywa się za pośrednictwem rejestrów SFR: TCON i TMOD

Układ transmisji szeregowej

Łącze szeregowe mikrokontrolera 8051 umożliwia prowadzenie synchronicznej lub asynchronicznej transmisji danych. Transmisja asynchroniczna jest transmisją full-duplex, natomiast synchroniczna jest transmisją half-duplex. Układ odbiornika posiada bufor odbiorczy, co pozwala na realizację procesu odbierania kolejnej danej przed pobraniem przez mikroprocesor danej już odebranej. Jest to jednak bufor jednobajtowy, więc nie odczytanie danej przez mikroprocesor przed końcem kolejnej transmisji powoduje utratę odebranego wcześniej bajtu. Podczas realizacji transmisji asynchronicznej nadawane dane wysyłane są linią TxD, zaś odbierane przez linię RxD. Podczas transmisji synchronicznej dane są odbierane i nadawane po linii RxD, a na linię TxD wysyłany jest sygnał taktujący. Do konfiguracji pracy układu transmisji szeregowej służy rejestr SCON z bloku SFR.

Układ przerwań

Mikrokontroler 8051 jest wyposażony w priorytetowy, dwupoziomowy układ przerwań. Układ przerwań jest specjalizowaną strukturą logiczną, której zadaniem jest monitorowanie stanu wskaźników przerwań i zgłaszanie faktu ustawienia określonego wskaźnika do układu sterowania. W mikrokontrolerze 8051 przerwanie może zostać wywołane przez jedno z pięciu wskaźników. Cztery ze wskaźników umieszczone są w rejestrze TCON. Piątym źródłem przerwania jest układ transmisji szeregowej. Przerwanie to jest zgłaszane przez ustawienie dowolnego z bitów RI lub TI rejestru SCON. W przypadku przerwań zewnętrznych i od układów czasowych, wskaźniki przerwania są sprzętowo zerowane po przyjęciu zgłoszenia przerwania (za wyjątkiem sytuacji, gdy przerwanie zewnętrzne jest zgłaszane niskim poziomem). Wskaźniki przerwania z układu transmisji szeregowej muszą być zerowane programowo przez procedurę obsługi przerwania, gdyż sprzętowe zerowanie uniemożliwiłoby

określenie, który ze wskaźników (RI czy TI) przerwanie wywołał. Do uaktywniania poszczególnych przerwań i określania ich priorytetów przeznaczone są rejestry sterujące IE i IP.

Generator sygnału taktującego

Mikrokontroler 8051 posiada wbudowany generator sygnału zegarowego, mogący współpracować z rezonatorem kwarcowym lub ceramicznym. Generator wytwarza sygnał taktujący mikrokontroler o częstotliwości równej częstotliwości zastosowanego rezonatora (układ powoduje wzbudzenie rezonatora na częstotliwości podstawowej). Przygotowanie generatora (a zarazem całego mikrokontrolera) do pracy polega na dołączeniu do wyprowadzeń XTAL1 i XTAL2 rezonatora.

Układ sterowania

Układ sterowania (wraz ze sterowaniem trybami uśpienia - obniżonego poboru mocy - tylko w 80C51) - najważniejszym zadaniem układu sterowania jest dekodowanie przesłanego z pamięci programu rozkazu i generacja na jego podstawie odpowiednich sygnałów sterujących pozostałymi elementami mikrokontrolera. Z układem sterowania współpracuje licznik rozkazów PC (16-bitowy rejestr zawierający adres kolejnego rozkazu przeznaczonego do wykonania) i rejestr rozkazów (rejestr przechowujący ostatnio pobrany rozkaz). Współpracuje z nim także układ przerwań (układ sterowania jest odpowiedzialny za sprzętową generację rozkazu LCALL wywołania procedury obsługi przerwania). Układ sterowania jest odpowiedzialny za generację następujących sygnałów zewnętrznych:

- $\overline{\text{PSEN}}$ - strob odczytu z zewnętrznej pamięci programu,
- $\overline{\text{RD}}$ - strob odczytu z zewnętrznej pamięci danych,
- $\overline{\text{WR}}$ - strob zapisu do zewnętrznej pamięci danych.
- ALE - sygnał sterujący buforem zatrzaskującym młodszy bajt adresu pamięci zewnętrznej.

Do układu sterowania są natomiast doprowadzone następujące sygnały zewnętrzne:

- $\overline{\text{EA}}$ - linia wyłączająca wewnętrzną pamięć programu.
- RST - linia zerowania mikrokontrolera.

Do komunikowania się mikrokontrolera z układami zewnętrznymi służą następujące magistrale:

1. **Magistrala danych -** jest to zbiór linii sygnałowych, którymi przesyłane są dane. Magistrala danych (ang. data bus) umożliwia przesyłanie danych między podzespołami mikroprocesora (głównie rejestrami), jak również między mikroprocesorem a pamięcią i urządzeniami zewnętrznymi, w dwu kierunkach – stąd określenie magistrala dwukierunkowa. Stan tej magistrali może być określony przez mikroprocesor (przy zapisie danych do urządzeń zewnętrznych lub pamięci) lub przez pamięć albo urządzenia zewnętrzne (przy odczycie). Zawiera ona najczęściej 4, 8, 16, 32 lub 64 linie, w zależności od tego ilu bitowy jest mikroprocesor.
2. **Magistrala adresowa -** jest to zbiór linii sygnałowych, które służą do wybierania przez mikroprocesor selektywnie komórek pamięci bądź urządzeń zewnętrznych. Magistrala adresowa (ang. address bus) jest magistralą jednokierunkową. Stan magistrali określa mikroprocesor. Ilość linii w magistrali adresowej określa obszar, jaki może zaadresować mikroprocesor, np. jeżeli magistrala jest 16-bitowa (o szerokości 16 linii) to obszar adresowania wynosi $2^{16} = 65536$.
3. **Magistrala sterująca -** jest to zbiór linii sygnałowych, po których przez mikroprocesor są przesyłane sygnały określające rodzaj operacji, jaką ma wykonać układ współpracujący (np. zapis lud odczyt do pamięci). Po magistrali sterującej (ang. control

bus) mikroprocesor otrzymuje również sygnały od urządzeń zewnętrznych informujące jego o ich stanie lub wykonywanej przez nie operacji (np. zgłoszenie przerwania). Dodatkowo przesyłane są po niej sygnały wysyłane przez użytkownika (np. reset). Ilość linii tej magistrali zależy od modelu mikroprocesora. Sygnały wysyłane nią są zarówno do jak i z mikroprocesora.

4.1.2. Pytania sprawdzające

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

1. Jakie są elementy architektury układu 8051?
2. Jakie znaczenie mają poszczególne bity w rejestrze flagowym układu 8051?
3. Jakie operacje może wykonywać arytmometr układu 8051?
4. Jak podzielona jest pamięć układu 8051?
5. Do czego służą rejestry specjalne i ile ich jest w układzie 8051?
6. Jaki sygnał i w jaki sposób układu 8051 steruje wyborem układu pamięci zewnętrznej?
7. Jak pogrupowane są linie we/wy mikrokontrolera 8051?
8. Z czego zbudowany jest i jak widziany jest przez mikroprocesor programowalny układ czasowy?
9. Do czego służy i jak działa układ transmisji szeregowej?
10. Jaki występuje układ przerwań w mikrokontrolerze 8051?
11. Co może wywołać przerwanie w mikrokontrolerze 8051?
12. Jakie zadania należą do układu sterowania?
13. Jakie sygnały zewnętrzne generuje układ sterowania?
14. Jakie sygnały doprowadzone są do układu sterowania?

4.1.3. Ćwiczenia

Ćwiczenie 1

Sygnały wejściowe i wyjściowe mikrokontrolera.

Celem ćwiczenia jest zapoznanie się z wejściowymi i wyjściowymi cechami mikrokontrolera 8051. Na podstawie kart katalogowych lub literatury mikrokontrolera 8051 określ, jakie występują w nim sygnały wejściowe i wyjściowe.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:

1) zorganizować stanowisko pracy do wykonania ćwiczenia,
2) zapoznać się z wyprowadzeniami mikrokontrolera 8051,
3) narysować wyprowadzenia mikrokontrolera 8051,
4) opisać znaczenie poszczególnych wyprowadzeń,

Wyposażenie stanowiska pracy:

- karty katalogowe układu 8051,
- schemat architektury układu 8051,
- poradnik dla ucznia,
- komputer z podłączeniem do Internetem.

Ćwiczenie 2

Celem ćwiczenia jest zapoznanie z zasadą działania rejestru flagowego, oraz arytmetyką binarną, dziesiętną i szesnastkową układu 8051.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:

1) zapoznać się z zasadą działanie rejestru flagowego układu 8051,
2) dokonać obliczeń,
3) wpisać znaczenie bitów w rejestrze flagowym,
4) wpisać wyniki do tabeli,
5) ocenić poprawność wykonywanego ćwiczenia.

Oblicz wartość wyrażenia i zaznacz wartości bitów w rejestrze flagowym:

a) AF(hex) + 33(hex)=
b) EF(hex) – A2(hex)=
c) 15(hex) * 03(hex)=
d) AC(hex) / 04(hex)=
e) 93(dec) + 45(dec)=
f) 71(dec) - 82(dec)=
g) 132(dec) / 12(dec)=
h) 15(dec) * 16(dec)=

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
	C	AC	F0	RS1	RS0	OV	-	P
a)								
b)								
c)								
d)								
e)								
f)								
g)								
h)								

Wyposażenie stanowiska pracy:

– komputer z podłączeniem do Internetem.

Ćwiczenie 3

Celem ćwiczenia jest zapoznanie się z mapą pamięci układu 8051.

Na podstawie na podstawie kart katalogowych i architektury układu 8051 narysuj mapę pamięci wewnętrznej układu z uwzględnieniem pamięci programu, pamięci danych oraz rejestrów specjalnych i portów.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:

1) zapoznać się z kartą katalogową układu 8051,
2) zapoznać się z architekturą układu 8051,
3) zapoznać się z rejestrami specjalnymi układu 8051,

4) sporządzić mapę pamięci wewnętrznej układu 8051 ,
5) ocenić poprawność wykonywanego ćwiczenia.

Wyposażenie stanowiska pracy:
– karty katalogowe układu 8051,
– schemat architektury układu 8051,
– poradnik dla ucznia,
– komputer z podłączeniem do Internetem.

Ćwiczenie 4

Celem ćwiczenia jest zapoznanie się z rejestrami specjalnymi i układami wewnętrznymi mikrokontrolera 8051.

Na podstawie na podstawie karty katalogowej mikrokontrolera 8051, opisu rejestrów specjalnych i układów wewnętrznych mikrokontrolera takich jak: porty wejścia-wyjścia, programowalny układ czasowy, układ transmisji szeregowej, układ przerwań, układ sterowania narysuj powiązania miedzy poszczególnymi układami i rejestrami specjalnymi.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) rozrysować w postaci bloków rejestry specjalne mikrokontrolera 8051,
2) narysować schemat blokowy mikrokontrolera 8051,
3) narysować powiązania poszczególnych układów z konkretnymi rejestrami specjalnymi,
4) ocenić poprawność wykonywanego ćwiczenia.

Wyposażenie stanowiska pracy:
– karty katalogowe układu 8051,
– schemat blokowy układu 8051,
– poradnik dla ucznia,
– komputer z podłączeniem do Internetem.

4.1.4. Sprawdzian postępów

Czy potrafisz:	**Tak**	**Nie**
1) wyróżnić bloki składowe mikrokontrolera 8051 ?	□	□
2) ustawić wartości bitów w rejestrze flagowym po operacji arytmetyczno-logicznej ?	□	□
3) narysować mapę pamięci wewnętrznej mikrokontrolera 8051 z uwzględnieniem jego rejestrów?	□	□
4) powiązać poszczególne rejestry specjalne z układami wewnętrznymi mikrokontrolera?	□	□
5) określić jak pracują poszczególne linie portów wejścia wyjścia?	□	□
6) opisać jak zbudowany jest programowalny układ czasowy i jak on działa?	□	□
7) określić jak pracuje układ transmisji szeregowej?	□	□
8) określić jak pracuje układ przerwań i jaką pełni role w mikrokontrolerze 8051?	□	□
9) określić za generacje jakich sygnałów odpowiedzialny jest układ sterowania?	□	□
10) określić jakie operacje może wykonywać arytmometr układu 8051 na argumentach ośmiobitowych?	□	□

4.2. Mikroprocesor, tryby pracy i współpraca z pamięcią

4.2.1. Materiał nauczania

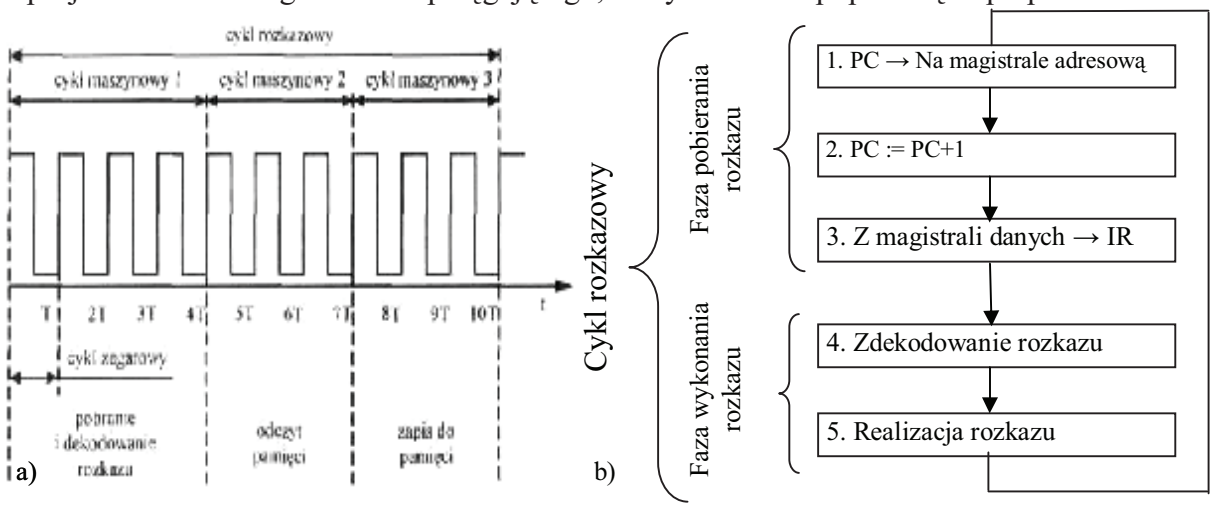
Cykl pracy mikroprocesora

Realizacja operacji wykonywanych przez mikroprocesor jest synchroniczna z sygnałami zegara taktującego sterującego pracą mikroprocesora. Generator ten może być elementem zewnętrznym (np. w mikroprocesorach 8080 i 8085) lub stanowić integralną cześć mikroprocesora (np. w mikrokontrolerach 8051) Najmniejszy przedział czasu, jaki jest stosowany w układzie sterującym, nosi nazwę cyklu zegarowego (rys. 5a).

Czas na potrzeby na przesłanie słowa binarnego (słowa danych, rozkazu) między mikroprocesorem i pamięcią lub urządzeniem we-wy nazywa się cyklem maszynowym (rys. 5a). Cykl maszynowy obejmuje kilka cykli zegarowych i może mieć różne długości zależne od rodzaju wykonywanych operacji.

Czas na potrzeby na pobranie i wykonanie jednego rozkazu nazywa się cyklem rozkazowym (rys.5b). Obejmuje on od jednego do kilku cykli maszynowych, zależnie od rodzaju rozkazu. Pierwszy cykl maszynowy w każdym cyklu rozkazowym jest związany z pobraniem rozkazu (Fetch).

Znajomość szczegółowych wykresów czasowych cyklu rozkazowego pozwala na poprawny dobór pamięci do szybkości mikroprocesora lub w przypadku niedopasowania, zaprojektowanie takiego układu sprzęgającego, który umożliwi poprawną współprace.



Rys.5. Cykl rozkazowy mikroprocesora:
a) przebieg czasowy [1 s.19]; b) schemat blokowy

Cykl rozkazowy przedstawiony w postaci schematu blokowego na rys. 5b przebiega w następujący sposób:

1. Przesłanie zawartości licznika adresu na magistrale adresową.
2. Ustalenie adresu następnej instrukcji w pewnej sekwencji stanowiącej program tzn. zwiększenie zawartości licznika rozkazów o 1.
3. Na podstawie adresu umieszczonego na magistrali adresowej przesłany jest rozkaz (z komórki pamięci o adresie ustawionym na magistrali adresowej) do rejestru rozkazów.
4. Określenie, który jest to rozkaz z listy rozkazów i wygenerowanie sygnałów sterujących umożliwiających jego wykonanie.
5. Wykonanie rozkazu może polegać na przetwarzaniu informacji zawartej w rejestrach mikroprocesora lub w pamięci lub przesłaniu informacji pomiędzy poszczególnymi blokami mikrokomputera.

Ponieważ sam mikroprocesor nie jest zdolny do samodzielnego funkcjonowania, dlatego wymaga dobudowy układów dodatkowych stanowiących system mikrokomputerowy.

Przez pojęcie układy otoczenia mikroprocesora określa się zwykle te części systemu mikroprocesorowego, które w sposób bezpośredni współpracują z jednostką centralną CPU. Są to więc: układy pamięci programu i danych oraz urządzenia wejściowe i wyjściowe. Przemysłowe sterowniki mikroprocesorowe, czyli tzw. mikrokontrolery, stosowane do sterowania bezpośredniego, należą do grupy małych systemów. Przy ich projektowaniu znaczną uwagę przywiązuje się do uproszczenia struktury i obniżki ceny jednostkowej. Dąży się również do maksymalnego uwzględnienia wymagań stawianych przez konkretny obiekt regulacji. W wyniku tych uproszczeń powstają struktury różniące się od rozwiązań spotykanych w mikrokomputerach ogólnego przeznaczenia.

Łączenie układów pamięci do systemu mikroprocesorowego

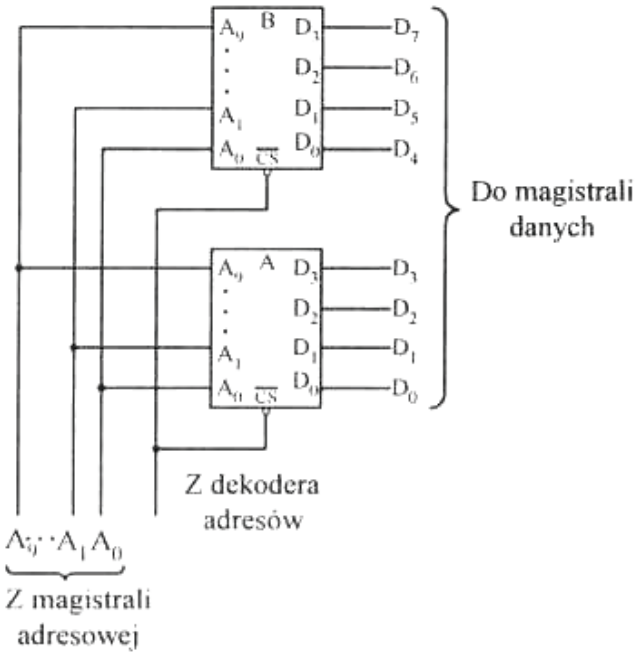
W zależności od potrzeb, z modułów scalonych pamięci jest budowana pamięć systemu mikroprocesorowego. Sposób budowy pamięci zależy od:

- typu posiadanych modułów, ich pojemności i organizacji;
- wymaganej mapy pamięci, czyli podziału przestrzeni adresowej mikrokomputera na części przeznaczone na pamięć typu ROM\ RAM i część niewykorzystaną.

Pamięć systemu mikroprocesorowego powinna spełniać następujące warunki:

- długość słowa pamięci powinna być równa długości słowa magistrali danych,
- komórki pamięci powinny być jednoznacznie adresowane tzn. jednemu adresowi odpowiada zawsze jedna i ta sama komórka pamięci.

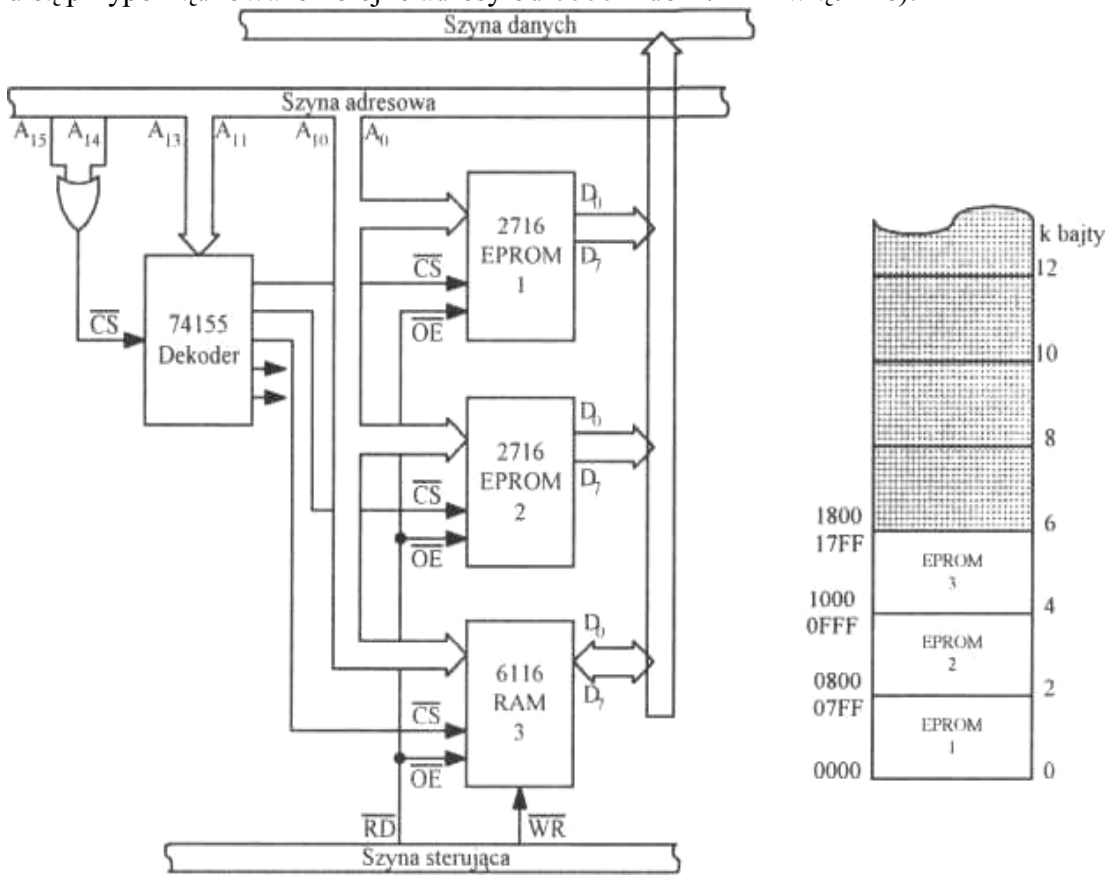
Dla zapewnienia odpowiedniej długości słowa, jeżeli moduły pamięci mają słowa krótsze, łączy się je równolegle. Na rys. 6 przedstawiono równoległe połączenie dwóch modułów pamięci o organizacji 1024x4 konieczne do uzyskania 8-bitowego słowa. Wejścia adresowe A i sterujące $\overline{CS}$ modułów są połączone równolegle, natomiast wyjścia modułu A tworzą bity 0-3, a wyjścia modułu B bity 4-7 słowa pamięci i będą podłączone do szyny danych mikrokomputera. W identyczny sposób można tworzyć słowa pamięci o dowolnej długości.



Rys.6. Przykład równoległego łączenia modułów pamięci [1 s.33]

Aby uzyskać jednoznaczność adresowania pamięci dzieli się magistralę adresową na dwie części: mniej znaczące bity adresu wybierają komórkę pamięci wewnątrz modułu i są dołączane do wejść A, bardziej znaczące bity adresu wybierają moduł. Wybieranie modułu pamięci na podstawie bardziej znaczących bitów adresu jest nazywane dekodowaniem adresów.

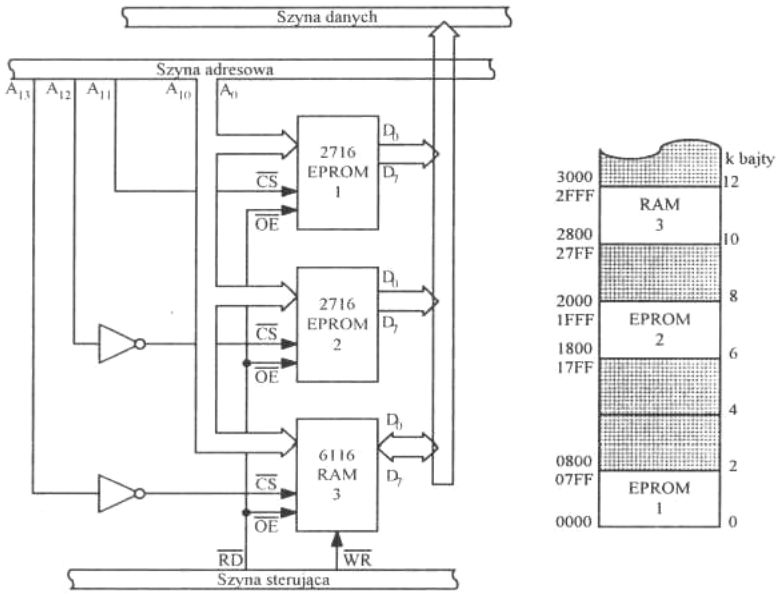
Dla przykładu na rys. 7 przedstawiono budowę bloku pamięci składającego się z modułów EPROM 2716 i RAM 6116, każdy o pojemności 2048 bajtów. Obok schematu połączeń umieszczono mapę pamięci, tzn. przyporządkowanie przestrzeni adresowej poszczególnym układom scalonym. Bity A_0-A_{10} magistrali adresowej są dołączone do wejść adresowych każdego z modułów i wybierają komórkę wewnątrz modułu. Bardziej znaczące bity magistrali adresowej A_{11}-A_{15} są podawane na dekoder, którego wyjścia są połączone z wejściami $\overline{CS}$ kolejnych modułów. W ten sposób w danej chwili będzie pracował tylko ten moduł, którego numer jest podany na liniach A_{11}-A_{15} magistrali adresowej. W przykładzie zastosowano tzw. pełne dekodowanie adresów, konieczne wtedy, gdy jest wykorzystywana cała przestrzeń adresowa (na mapie pamięci kolejnym bajtom informacji przechowywanej w bloku są przyporządkowane kolejne adresy od 0000H do 17FFH włącznie).



Rys.7. Przykład równoległego łączenia modułów pamięci i mapa pamięci przy pełnym dekodowaniu adresów [1 s.34]

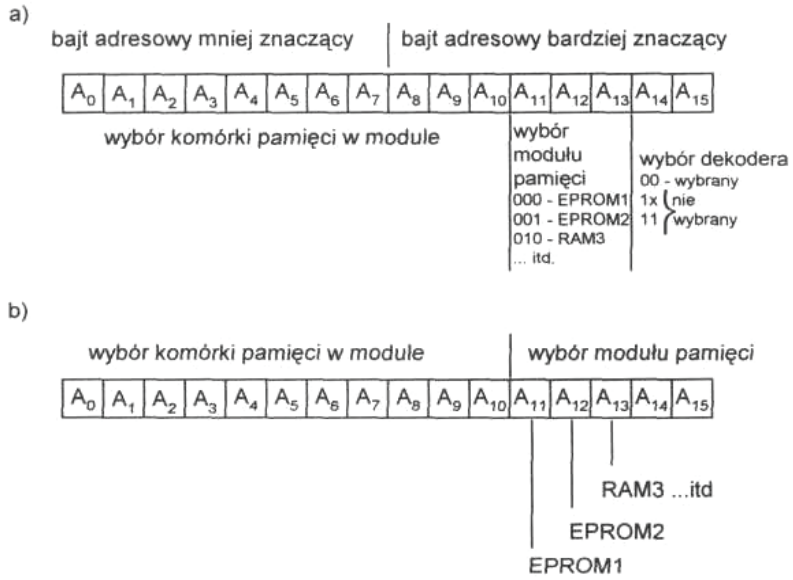
Gdy pojemność pamięci niezbędnej w systemie mikroprocesorowym jest mniejsza niż przestrzeń adresowa, można zastosować dekodowanie częściowe.

W układzie jak na rys.8 zastosowano tzw. selekcją linową. Poszczególnym układom scalonym przyporządkowano kolejne starsze bity adresowe A_{11}, A_{12}, A_{13}, eliminując potrzebę zastosowania dekodera. W tym przypadku mapa pamięci traci ciągły charakter. W bloku wykorzystuje się adresy od 0000H do 07FFH, od 1800H do 1FFFH oraz od 2800H do 2FFFH włącznie. Pozostałe części pamięci zostają utracone. Sposób selekcji liniowej stosuje się w małych, przemysłowych systemach mikroprocesorowych w celu obniżenia kosztów urządzeń.



Rys.8. Przykład łączenia modułów pamięci i mapa pamięci przy częściowym dekodowaniu adresów (selekcja liniowa) [1 s.35]

W zależności od przyjętego sposobu selekcji modułów pamięci różny jest format słowa adresowego. Selekcji z pełnym dekodowaniem odpowiada format słowa przedstawiony na rys. 9a. Bity od A_0 do A_{10} służą do wyboru adresu wewnątrz modułów. Ta część słowa jest dekodowana przez wewnętrzne selektory wierszy i kolumn w każdym module pamięci. Bity A_{11}, A_{12}, A_{13} są dekodowane przez scalony dekoder 74155. Możliwe jest więc odróżnienie ośmiu modułów, z których każdy ma pojemność 2 kB. Najbardziej znaczące bity A_{14} i A_{15} można wykorzystać do wyboru jednego z czterech równorzędnych dekoderów (za pomocą dodatkowego dekodera nadrzędnego). Można więc docelowo wybierać jeden z 32 modułów pamięci o pojemności 2 kB, co wypełnia całą możliwą przestrzeń mikroprocesora 8-bitowego (tzn. 64 kilobajty). W przypadku selekcji liniowej ulega zmianie sposób dekodowania pięciu najbardziej znaczących bitów słowa adresowego (rys. 9b). Każdemu modułowi pamięci jest przyporządkowany jeden z bitów A_{11}-A_{15}. Wobec tego istnieje możliwość wyboru jednego z pięciu modułów pamięci o pojemności 2 kB, co oznacza możliwość wykorzystania jedynie części przestrzeni adresowej o pojemności 10 kB.

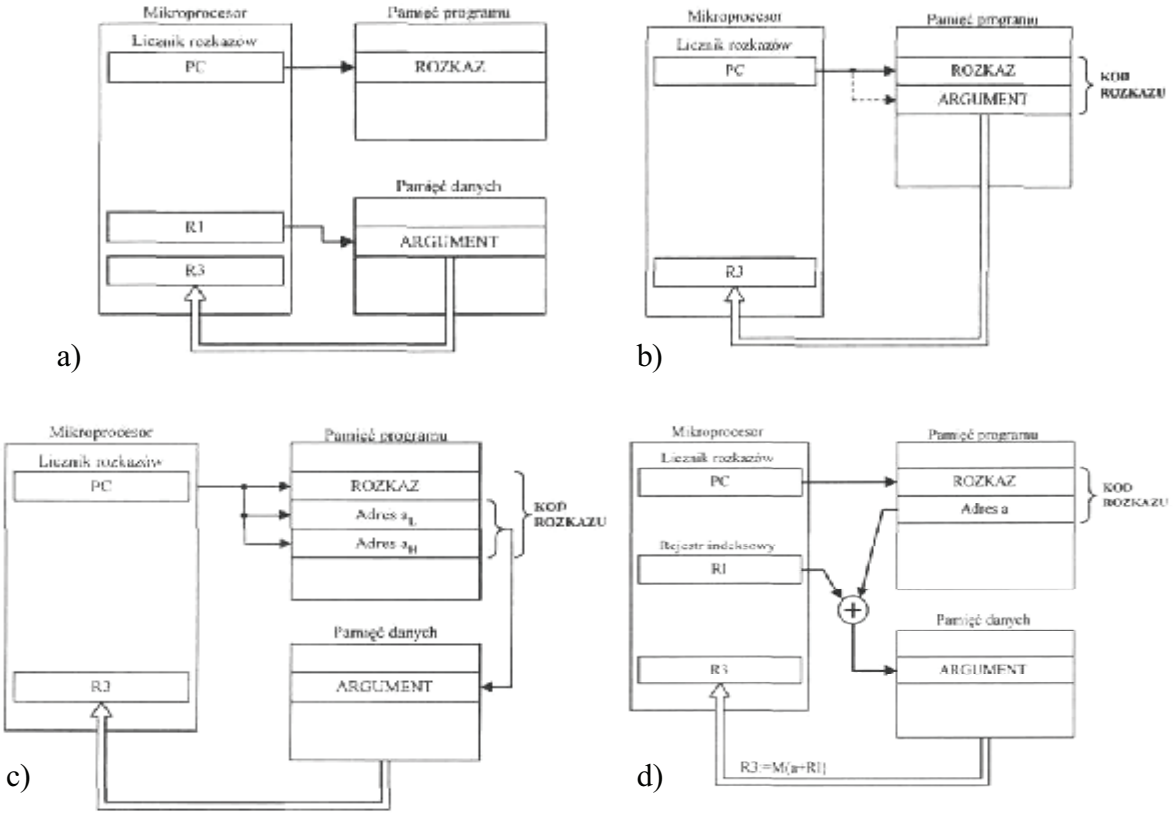


Rys.9. Formaty słów adresowych pamięci dla przykładów z rys. 7, 8 a) selekcja z pełnym dekodowaniem, b) selekcja liniowa [3. s.36]

Podstawowe tryby adresowania

Przesyłanie informacji między rejestrami mikroprocesora oparte jest o tzw. rozkaz bezadresowy. Jego wykonanie nie wymaga odwoływania się do modułów zewnętrznych. Przesyłanie informacji natomiast między rejestrem a pamięcią wymaga tzw. rozkazu adresowego i dlatego konieczne jest określanie adresu efektywnego komórki pamięci. Wiąże się to z tzw. trybami adresowania pamięci (ang. addressing mode). Tryb adresowania określa miejsce, gdzie umieszczany jest adres argumentu lub sposób w jaki jest on obliczany.

Adresowanie pośrednie (ang. indirect addressing) odnosi się do rozkazów zawierających poza kodem operacji adres komórki pamięci, w której znajduje się adres argumentu. Odmianą adresowania pośredniego jest tzw. adresowanie za pomocą wskaźników (ang. pointer addressing) zwane inaczej adresowaniem rejestrowym pośrednim lub adresowaniem zawartością rejestrów. Odnosi się ono do rozkazów, które swoim kodem wskazują rejestr lub rejestry zawierające adres argumentu rozkazu. Rejestry te nazywa się rejestrami wskaźnikowymi lub licznikiem danych. Na zawartości rejestru wskaźników można wykonywać operacje arytmetyczne (np. dodawanie jedności) czyli modyfikować adres. Możliwość modyfikacji adresu jest ważną cechą trybu adresowania, ułatwia bowiem wykonywanie operacji na złożonych strukturach danych. Na rysunku 10a przedstawiono schematycznie adresowanie za pomocą wskaźnika. Przykładowo w fazie wykonania, na magistralę adresową jest wysyłana zawartość rejestru Rl, a słowo z pamięci (zwane argumentem, operandem) jest podawane na magistralę danych i wpisywane do R3.



Rys.10. Tryby adresowania układów pamięci:
a) adresowanie za pomocą wskaźnika [1 s.26]; b) adresowanie natychmiastowe [1 s.26]; c) adresowanie bezpośrednie [1 s.27]; d) adresowanie indeksowe [1 s.28];

Adresowanie natychmiastowe (ang. immediate addressing) przedstawione na rys. 10b. W tym przypadku argument (np. stała) jest umieszczony w pamięci programu bezpośrednio za kodem rozkazu, czyli po pobraniu kodu rozkazu adres argumentu jest zawarty w liczniku rozkazów. Rozkazy o adresowaniu natychmiastowym nazywa się rozkazami z argumentem

bezpośrednim. W fazie wykonywania rozkazu zawartość licznika rozkazów PC jest automatycznie zwiększana o 1 (podobnie jak po pobraniu kodu operacji), tak aby po wykonaniu licznik rozkazów zawierał adres następnego rozkazu.

Adresowanie bezpośrednie (ang. direct addressing) przedstawiono na rys. 10c. Przy takim adresowaniu adres argumentu jest umieszczany w pamięci programu, w słowie następującym za rozkazem. Szesnastobitowy adres (mikroprocesorów 8 bitowych) zajmuje dwa kolejne słowa pamięci. W pierwszym jest umieszczana mniej znacząca część adresu $\mathbf{a_L}$ (bity 0-7), a w drugim - bardziej znacząca $\mathbf{a_H}$ (bity 8-15). Adresowanie bezpośrednie jest często stosowane w rozkazach skoku oraz do adresowania danych zajmujących pojedyncze słowa pamięci.

Adresowaniu indeksowe (ang. index addressing) adres argumentu otrzymuje się przez dodanie adresu bezpośredniego, umieszczonego za rozkazem, do zawartości rejestru procesora np. **RI** (wskaźnika danych), zwanego w tym przypadku rejestrem indeksowym. Schemat adresowania indeksowego przedstawiono na rys. 10d. Jak widać, adresowanie indeksowe jest połączeniem adresowania zawartością rejestru z adresowaniem bezpośrednim. Istnieje tu możliwość łatwej modyfikacji adresu.

4.2.2. Pytania sprawdzające

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

1. Co nazywamy cyklem rozkazowym?
2. Co nazywamy cyklem maszynowym?
3. Na czym polega organizacja pamięci?
4. Od czego zależy sposób budowy pamięci?
5. Jakie warunki powinna spełniać pamięć systemu mikroprocesorowego?
6. Co to jest mapa pamięci?
7. Jak podłączana jest pamięć do układu mikroprocesorowego?
8. Co nazywamy trybami adresowania pamięci?
9. Na czym polega adresowanie pośrednie?
10. Na czym polega adresowanie bezpośrednie?
11. Na czym polega adresowanie natychmiastowe?
12. Na czym polega adresowanie indeksowe?

4.2.3. Ćwiczenia

Ćwiczenie 1

Parametry układów pamięci.

Na podstawie kart katalogowych wybranych układów pamięci określ, jakie posiadają parametry, organizacje, oraz jakie występują miedzy nimi różnice.

Celem ćwiczenia jest zapoznanie się z kartami katalogowymi i parametrami wybranych układów pamięci RAM i ROM.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:

1) zapoznać się z oznaczenia i parametrami układów pamięci,
2) zorganizować stanowisko pracy do wykonania ćwiczenia,
3) zapoznać się z kartami katalogowymi układów pamięci,
4) wyszukać różnice pomiędzy poszczególnymi układami pamięci.
5) ocenić poprawność wykonywanego ćwiczenia.

Wyposażenie stanowiska pracy:
– karty katalogowe układów pamięci,
– komputer z podłączeniem do Internetem.

Ćwiczenie 2

Celem ćwiczenia jest zapoznanie się z trybami adresowania pamięci.
Na podstawie przedstawionych w materiale nauczania trybów adresowania pamięci, określ różnice między nimi, oraz jakie posiadają wady i zalety.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zapoznać się z sposobami adresowania pamięci,
2) wypisać różnice między nimi, oraz wady i zalety poszczególnych trybów adresowania.
3) ocenić poprawność wykonywanego ćwiczenia.

Wyposażenie stanowiska pracy:
– poradnik dla ucznia,
– komputer z podłączeniem do Internetem.

Ćwiczenie 3

Celem ćwiczenia jest zapoznanie się podłączaniem pamięci do mikroprocesora oraz tworzeniem mapy pamięci układu mikroprocesorowego.

Na podstawie kart katalogowych układów pamięci zaprojektuj moduł pamięci składający się z pamięci dwóch modułów pamięci ROM o organizacji 2k x 4, oraz ośmiu układów pamięci statycznej RAM o organizacji 16k x 1. Narysuj sposób podłączenia wyprowadzeń adresowych i danych do magistral oraz mapę pamięci.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) odnaleźć karty katalogowe układów pamięci o zadanej organizacji,
2) narysować sposób podłączenia wyprowadzeń poszczególnych układów do magistrali adresowej i danych oraz odpowiednich sygnałów sterujących,
3) sporządzić mapę pamięci zaproponowanego modułu ,
4) ocenić poprawność wykonywanego ćwiczenia.

Wyposażenie stanowiska pracy:
– poradnik dla ucznia,
– karty katalogowe układów pamięci,
– komputer z podłączeniem do Internetem.

Ćwiczenie 4

Celem ćwiczenia jest zapoznanie się podłączaniem pomięci do mikroprocesora oraz mapą pamięci.
Na podstawie na podstawie kart katalogowych układów pamięci zaprojektuj moduł pamięci składający się z dowolnej ilości układów pamięci w taki sposób aby uzyskać organizacje: ROM 2k x 8 i RAM 32k x 8. Do budowy modułu pamięci możesz użyć dowolnych układów. Narysuj sposób podłączenia wyprowadzeń adresowych i danych do magistral oraz mapę pamięci.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) odnaleźć karty katalogowe układów pamięci,
2) wybrać układy z których chcesz zaprojektować moduł pamięci,
3) narysować sposób podłączenia wyprowadzeń poszczególnych układów do magistrali adresowej i danych oraz odpowiednich sygnałów sterujących,
4) sporządzić mapę pamięci zaproponowanego modułu ,
5) ocenić poprawność wykonywanego ćwiczenia.

Wyposażenie stanowiska pracy:
– karty katalogowe układów pamięci,
– poradnik dla ucznia,
– komputer z podłączeniem do Internetem.

4.2.4 Sprawdzian postępów

Czy potrafisz:	**Tak**	**Nie**
1) określić co to jest cykl rozkazowy?	□	□
2) określić co to jest cykl maszynowy?	□	□
3) określić co to jest organizacja pamięci?	□	□
4) obliczyć pojemność pamięci na podstawie znajomości jej organizacji?	□	□
5) zaprojektować moduł pamięci o określonych parametrach?	□	□
6) narysować mapę pamięci wybranego modułu pamięci?	□	□
7) narysować sposób podłączenia układów pamięci do systemu mikroprocesorowego?	□	□
8) opisać tryby adresowania układów pamięci?	□	□
9) podać różnice między sposobami adresowania pamięci?	□	□
10) posłużyć się Internetem w celu wyszukania karty katalogowej danego układu?	□	□

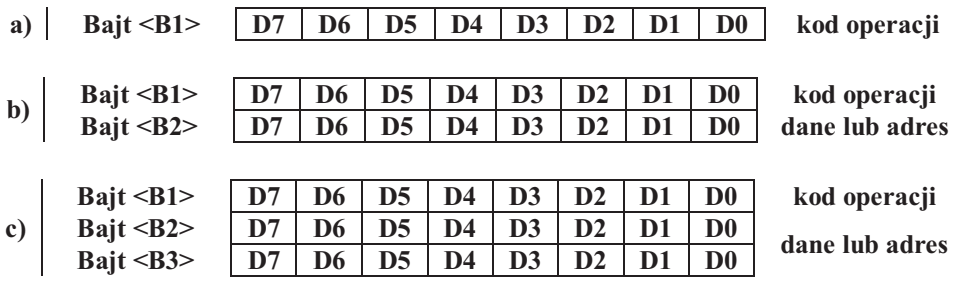
4.3. Asembler mikrokontrolera 8051. Operacje przesyłania danych.

4.3.1. Materiał nauczania

Asemblery lub **języki asemblerów** (ang. assembly languages) to w informatyce rodzina języków programowania niskiego poziomu, w których zasadniczo jedno polecenie odpowiada jednemu rozkazowi procesora. Są to języki powstałe na bazie języka maszynowego poprzez zastąpienie liczb odpowiadających fragmentom rozkazów kodu maszynowego ich symbolicznymi odpowiednikami (mnemonikami). Dzięki zamianie liczb na tzw. mnemoniki można pisać programy w miarę zrozumiałe dla człowieka, a jednocześnie bezpośrednio tłumaczone na kod maszynowy procesora, co pozwala zapewnić duży stopień kontroli programisty nad zachowaniem procesora.
Asembler jest również programem, który tłumaczy kod źródłowy programu (zapisany w asemblerze danego mikroprocesora) na postać binarną (zestaw bitów). Taki proces tłumaczenia nosi nazwę **asemblacji**. W wyniku asemblacji program zapisywany jest w pliku w formacie umożliwiającym jego wpisanie do pamięci programu. W czasie asemblacji powstaje również plik tekstowy zawierający kod źródłowy programu uzupełniony o różne informacje wytworzone przez asembler (numery linii, kody rozkazów, komunikaty o błędach itp.). plik jest ten przeznaczony dla programisty i umożliwia mu wyszukanie i usunięcie błędów.

Struktura rozkazów mikrokontrolera 8051

Lista rozkazów mikrokontrolerów rodziny '51 zawiera 111 instrukcji: 49 jednobajtowych, 45 dwubajtowych i 17 trzybajtowych. Struktury rozkazów w mikrokontrolerze 8051 przedstawiono na rys. 11.



Rys.11. Struktura rozkazów w mikrokontrolerze 8051:
a) jednobajtowe; b) dwubajtowe; c) trzybajtowe

Stałe liczbowe

Stała liczbowa musi zaczynać się od cyfry. Asembler mikrokontrolera 8051 akceptuje następujące typy stałych liczbowych zestawione w tabeli 3:

Tabela 3 Typy stałych liczbowych

Typ	**Składnia**	**Przykład**
Dziesiętny	< cyfry >	125
Szesnastkowy	< cyfry szesnastkowe > H	OFFFh
Ósemkowy	< cyfry ósemkowe > O	7777o
Binarny	< cyfry binarne > B	10101b
Znakowy	'<znak>'	'Z'

Format linii w Asemblerze

Typowa linia programu w asemblerze wygląda następująco:

[<**etykieta**>] [<**rozkaz**>] [<**operandy**>] [;<**komentarz**>]

Znaczenie poszczególnych pól linii programu jest następujące:

<**etykieta**> - symbol umieszczony na samym początku linii (pierwszy znak etykiety musi być pierwszym znakiem w linii). Etykieta musi zaczynać się od litery lub znaku podkreślenia'_', i może zawierać dowolną kombinację liter, cyfr i podkreśleń. Jeśli etykieta jest zakończona dwukropkiem to nadawana jest jej wartość określająca jej pozycję w kodzie źródłowym (adres rozkazu z tej linii programu). Etykiety (symbole) stosowane z dyrektywami nadającymi im wartość nie są zakończone dwukropkiem.

<**rozkaz**> - mnemonik kodu maszynowego procesora, dyrektywa asemblera lub makro.

<**operandy**> - informacje wymagane przez mnemonik, dyrektywę asemblera lub makro. Poszczególne operandy są oddzielane przecinkami.

<**komentarz**> - wszystkie znaki występujące po średniku (;) są traktowane jako komentarz i ignorowane przez asembler.

Poszczególne pola linii programu muszą być oddzielone między sobą, co najmniej jednym znakiem spacji (lub tabulacji). W programie mogą występować puste linie lub linie zawierające wyłącznie komentarz.

W liście rozkazów mikrokontrolera 8051 stosuje się następujące oznaczenia:

- Rn - rejestry R0-R7.
- direct - adres wewnętrznej pamięci danych (128B) i rejestry SFR.
- @Ri - adres pośredni w rejestrze indeksowym R0 lub R1.
- bit - bity pamięci danych i SFR adresowalne bezpośrednio.
- #data - stała 8-bitowa.
- #data16 - stała 16-bitowa.
- addr16 - adres 16-bitowy (obszar 64kB).
- addr11 - adres 11-bitowy (w stronie 2kB).
- rel - adres względny -128...+127 (nazwa etykiety).
- /bit - negacja bitu.

Przykłady programów

Przedstawione programy są pisane w sposób ogólny związany z listą rozkazów mikrokontrolera 8051. W celu przetestowania ich i wpisania do mikrokontrolera należy użyć dostępnych na zajęciach Dydaktycznym Systemem Mikroprocesorowych (DSM), skompilować je na ich formaty oraz zmodyfikować wpisując procedury i rozkazy umożliwiające ich podejrzenie na np. wyświetlaczach LDC danego systemu. Należy zapoznać się z instrukcja obsługi DSM dostępną przy stanowisku ucznia.

Program wpisujący liczby do akumulatora

Analizując ten program zapoznasz się, w jaki sposób można wpisywać wartości liczbowe do akumulatora.

```
        LJMP    START
        ORG     100H
START:
        MOV     A,#10H          ;wpisz liczbę 10H do A
        MOV     ACC,#20H        ;wpisz liczbę 20H do ACC
        LJMP    $               ; pozostań w tej linii
```

Program ten polega na wpisaniu dwóch liczb do akumulatora najpierw 10H a następnie zastąpienie jej przez 20H. Różnica w oznaczeniu Akumulatora polega na tym, że raz traktowany jest jako akumulator A, a za drugim razem jako rejestr znajdujący się w obszarze

rejestrów specjalnych i oznaczamy do jako ACC. Wykonanie drugiego rozkazu trwa półtora razy dłużej.
Znaczenia użytych rozkazów.

MOV Rn,#dana ;instrukcja załadowania 8-bitowej liczby „dana" do rejestru w tym wypadku akumulatora

LJMP - rozkaz skoku w obszarze 64kB.

Działanie: wpisuje do licznika rozkazów podany adres. Dozwolone argumenty: addr16 – 16-bitowy adres w pełnej przestrzeni adresowej 64kB pamięci programu.

ORG - ustawienie adresu dla następnego bloku kodu – dyrektywa sterująca

Składnia:
ORG <wyrażenie>
Ustawienie adresu dla następującego po tej dyrektywie bloku kodu. Adres dla następnej instrukcji procesora jest ustalany poprzez wyliczenie wartości wyrażenia. Możliwe jest jedynie zwiększanie aktualnego adresu kodu. Próba zmniejszenia adresu jest sygnalizowana jako błąd. Standardowo kod programu jest umieszczany rozpoczynając od adresu 0.
Znak '#' w mnemoniku poprzedza argument bezpośredni – konkretną wartość liczbową
Znak '$' jest interpretowany jako bieżący adres.
Składnia: Rozkaz_Skoku $

Program adresujący bity akumulatora

W programie tym zapoznasz się, w jaki sposób można ustawiać wartości poszczególnych bitów w rejestrach w tym przypadku w akumulatorze.

```
        LJMP   START
        ORG    100H
START:
        MOV    A,#32H       ;wpisz liczbę do A
        SETB   ACC.0        ;ustaw bit 0 akumulatora na 1 w akumulatorze znajduje się
                            liczba 33H
        MOV    R1,A         ;wpisz liczbę do R1 liczbę z A,  R1 ← A=33H
        SETB   ACC.2        ;ustaw bit 2 akumulatora na 1 w akumulatorze znajduje się
                            liczba 37H
        MOV    R2,A         ;wpisz liczbę do R2 liczbę z A,  R2 ← A=37H
        CLR    ACC.4        ;ustaw bit 4 akumulatora na 0 w akumulatorze znajduje się
                            liczba 27H
        MOV    R3,A         ;wpisz liczbę do R3 liczbę z A,  R3 ← A=27H
        LJMP   $            ; pozostań w tej linii
```

Rozkaz

SETB bit ;ustawiony zostaje bit na 1, którego adres podany jest bezpośrednio
CLR bit ;ustawiony zostaje bit na 0, którego adres podany jest bezpośrednio

Program wykorzystujący adresowanie bezpośrednie przy zapisie danych do rejestrów i adresowanie rejestrowe przy wpisywaniu wartości rejestrów do akumulatora

W programie tym zapoznasz się, w jaki sposób można wpisywać dane do rejestrów, aktywować poszczególne banki rejestrów poprzez ustawienia bitów wyboru wykorzystując adresowanie bezpośrednie i rejestrowe oraz w jaki sposób definiuje się dane.

```
B0R7    EQU   7H        ;rejestr R7 z banku 0
B1R7    EQU   8H+7H     ;rejestr R7 z banku 1
B2R7    EQU   10H+7H    ;rejestr R7 z banku 2
B3R7    EQU   18H+7H    ;rejestr R7 z banku 3
    LJMP  START
```

```
        ORG     100H
START:
        MOV     B0R7,#0         ;wpisz numer banku 0 do rejestru R7
        MOV     B1R7,#1         ;wpisz numer banku 1 do rejestru R7
        MOV     B2R7,#2         ;wpisz numer banku 2 do rejestru R7
        MOV     B3R7,#3         ;wpisz numer banku 3 do rejestru R7 aktywny bank 0
        MOV     A,R7            ;A ← R7=0
        SETB    RS0             ; aktywny bank 1 przełączenie banku
        MOV     A,R7            ;A ← R7=1
        SETB    RS1             ; aktywny bank 3 przełączenie banku
        MOV     A,R7            ;A ← R7=3
        CLR     RS0             ; aktywny bank 2 przełączenie banku
        MOV     A,R7            ;A ← R7=2
        LJMP    $               ; pozostań w tej linii
```

Użyte w programie rozkazy:

EQU - definiowanie stałej – dyrektywa danych

Składnia:

<symbol> EQU <wyrażenie>

Symbolowi <symbol> przypisywana jest wartość wyrażenia. Typ symbolu ustalany jest na podstawie wyrażenia. Każda wartość zdefiniowana dyrektywą EQU jest stałą i nie może być zmieniana w trakcie asemblacji.

Znak „+” w tym wypadku oznacza dodawanie arytmetyczne liczb szesnastkowych tworząc adres bezpośredni rejestru

;Program wykorzystujący adresowanie pośrednie przy zapisie danych do rejestrów

W programie tym zapoznasz się, w jaki sposób można wpisywać dane do rejestrów, przy adresowaniu pośrednim oraz na jakiej zasadzie działa pętla. Program ten polega na wpisaniu do 10 komórek pamięci o adresach od 40H do 49H liczby 0. Działanie pętli polega na tym, że program powtarza daną część zaczynającą się od nazwy etykiety (PETLA) tak długo aż osiągnięty zostanie warunek, w tym wypadku wartość rejestru R2 wynosząca 0. Po spełnieniu tego warunku procesor przejdzie do wykonanie następnej linii programu.

```
        LJMP    START
        ORG     100H
START:
        MOV     43H,#55H        ;wpisz 55 do komórki pamięci o adresie 43H,
                                (43H) ← #55H
        MOV     A,43H           ;do akumulatora przepisz zawartość komórki pamięci 43H
                                 A← (43H) = 55H
        MOV     R0,#40H         ;do R0 wpisz liczbę 40H która będzie adresem
        MOV     R2,#10          ;do R2 wpisz liczbę 10 - licznik pętli
PETLA:                          ;zeruj 10 komórek pamięci od adresu 40H, czyli obszar
                                od 40H do 49H, PETLA-nazwa etykiety
        MOV     @R0,#0          ;wpisz liczbę 0 pod adres komórki umieszczony w R0
        INC     R0              ;zwiększ wartość R0 o 1, tutaj adres komórki pamięci
        DJNZ    R2,PETLA        ;powtórz zapis n-razy zgodnie z licznikiem do momentu
                                 aż wartości rejestrze R2 nie osiągnie zera
        MOV     A,43H           ;A ← (43H)=0
        LJMP    $               ; pozostań w tej linii
```

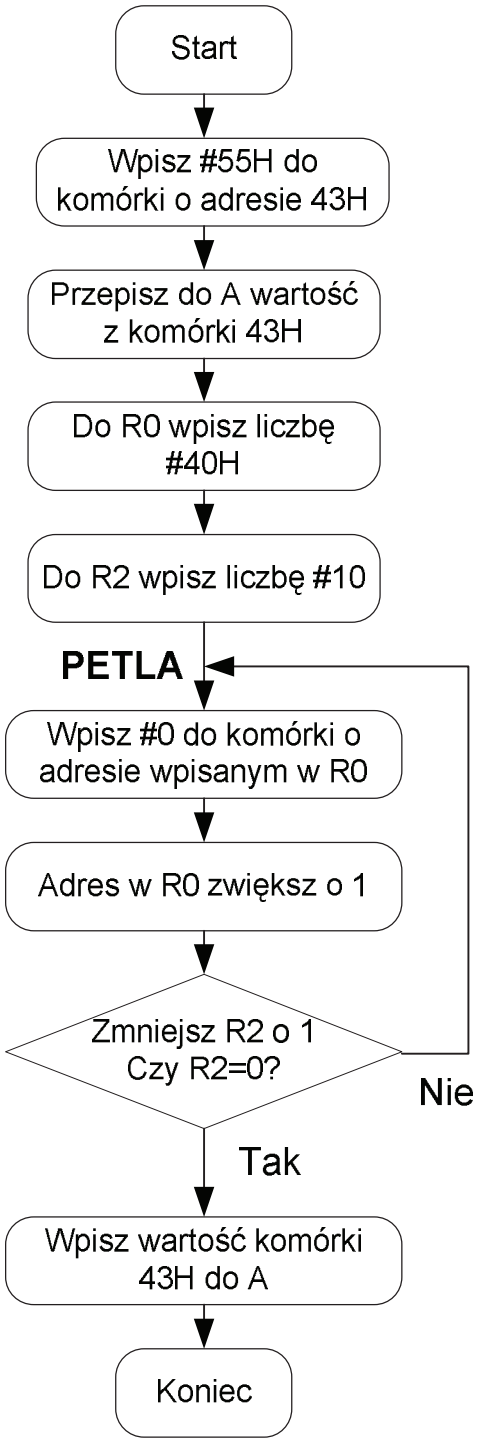
Użyte w programie rozkazy:

INC Rn **;**do zawartości rejestru Rn dodawana jest 1, Rn← Rn+1

DJNZ Rn, rel ;zmniejszona zostaje wartość w rejestrze Rn (R0...R7) o jeden, a następnie, ;jeżeli wartość w Rn nie jest równa zero, to następuje skok do początku etykiety.

Znak '@' przed oznaczeniem rejestru oznacza, że w rejestrze zapisany jest adres komórki pamięci

Program wykorzystujący adresowanie pośrednie przy zapisie danych do rejestrów w postaci algorytmu działania pokazano rys. 12. Rysowanie algorytmów przy programowaniu ułatwia zrozumienie zasady działania programu oraz ułatwia samo programowanie.



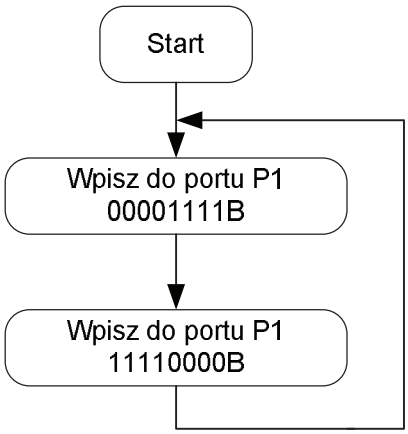
Rys.12. Algorytm programu wykorzystującego adresowanie pośrednie przy zapisie danych do rejestrów

Program ustawianie linii w porcie mikrokontrolera 8051

W programie tym zapoznasz się, w jaki sposób można zmieniać stan linii w porcie mikrokontrolera oraz jak działa pętla bez zakończenia. Program przedstawiony poniżej wpisuje na zmianę do portu P1 dwie stałe P1_0 i P1_1 przełączając wszystkie linie w porcie. Będzie wykonywany tak długo aż nie zostanie przerwany np. sygnałem reset.

```
P1_0    EQU    00001111B
P1_1    EQU    11110000B
        LJMP   START
        ORG    100H
START:
PORT:                          ;pętla zmiany stanu linii w porcie P1
        MOV    P1,# P1_0       ;wpisz 00001111B do portu P1
        MOV    P1,# P1_1       ;wpisz 11110000B do portu P1
        LJMP   PORT            ;powtórz procedurę PORT
```

Program ustawianie linii w porcie mikrokontrolera 8051 w postaci algorytmu działania pokazano rys. 13.



Rys.13. Algorytm programu zmieniającego stan linii w porcie P1 mikrokontrolera 8051

4.3.2. Pytania sprawdzające

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

1. Co to jest asembler?
2. Co to jest mnemonik?
3. Jak wygląda struktura rozkazów?
4. Jak zbudowana jest linia programu w asemblerze?
5. Co oznacza etykieta?
6. Co to są operandy?
7. Co to jest komentarz?
8. Do czego służy rozkaz MOV i jak on jest zbudowany?
9. Co oznacza komenda ORG i do czego służy?
10. Do czego służą rozkazy SETB i CLR i czym się różnią?
11. Do czego wykorzystuje się komendę EQU?
12. Do czego służą znaki # i @ w asemblerze?
13. Jak działa rozkaz DJNZ?
14. Do czego służy algorytm programu i co przedstawia?

4.3.3. Ćwiczenia

Ćwiczenie 1

Celem ćwiczenia jest poznanie zasad tworzenia algorytmów. Na podstawie programu adresującego bity akumulatora narysuj jego algorytm.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zapoznać się z programem adresującym bity akumulatora,
2) zapoznać się z listą rozkazów mikrokontrolera 8051,
3) narysować algorytm programu,
4) ocenić poprawność wykonywanego ćwiczenia.

Wyposażenie stanowiska pracy:
- karty katalogowe układu 8051,
- lista rozkazów mikrokontrolera 8051,
- poradnik dla ucznia,
- komputer.

Ćwiczenie 2

Celem ćwiczenia jest zapoznanie z sposobem programowania poszczególnych bitów w rejestrach. Napisz program zmieniający bity w akumulatorze, aby otrzymać następujące liczby szesnastkowe począwszy od liczby AAH: 2AH, 3AH, 3BH, 1BH, 9BH, wpisując wyniki poszczególne wyniki operacji do komórek pamięci o adresie od 40H do 44H. Narysuj algorytm programu.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zapoznać się z programem adresującym bity akumulatora,
2) zapoznać się z listą rozkazów mikrokontrolera 8051,
3) narysować algorytm programu,
4) napisać program,
5) ocenić poprawność wykonywanego ćwiczenia.

Wyposażenie stanowiska pracy:
- lista rozkazów mikrokontrolera 8051,
- komputer,
- przewodnik dla ucznia,
- dydaktyczny system mikroprocesorowy,
- instrukcja programowania i obsługi dydaktycznego systemu mikroprocesorowego,

Ćwiczenie 3

Celem ćwiczenia jest zapoznanie z sposobem programowania wykorzystującym adresowanie pośrednie jak i działaniem pętli. Napisz program wpisujący kolejno liczby od #10H do 1EH do 15 komórek pamięci od komórki adresie 40H do 54H. Narysuj algorytm programu.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zapoznać się z programami pokazanymi w materiale do nauki,
2) zapoznać się z listą rozkazów mikrokontrolera 8051,
3) narysować algorytm programu,
4) napisać program,
5) ocenić poprawność wykonywanego ćwiczenia.

Wyposażenie stanowiska pracy:
– lista rozkazów mikrokontrolera 8051,
– komputer,
– przewodnik dla ucznia,
– dydaktyczny system mikroprocesorowy,
– instrukcja programowania i obsługi dydaktycznego systemu mikroprocesorowego,

Ćwiczenie 4

Celem ćwiczenia jest zapoznanie z sposobem ustawiania wartosci linii w porcie. Napisz program ustawiający linie w porcie P2 w taki sposób, aby otrzymać następujące ich stany 11000000B, 00110000B, 00001100B i 00000011B. Narysuj algorytm programu.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zapoznać się z programami pokazanymi w materiale do nauki,
2) zapoznać się z listą rozkazów mikrokontrolera 8051,
3) narysować algorytm programu,
4) napisać program,
5) ocenić poprawność wykonywanego ćwiczenia.

Wyposażenie stanowiska pracy:
– lista rozkazów mikrokontrolera 8051,
– komputer,
– przewodnik dla ucznia,
– dydaktyczny system mikroprocesorowy,
– instrukcja programowania i obsługi dydaktycznego systemu mikroprocesorowego,

4.3.4 Sprawdzian postępów

Czy potrafisz:	**Tak**	**Nie**
1) narysować strukturę rozkazów mikrokontrolera 8051?	□	□
2) zapisać liczby w postaci jaką przyjmuje asembler ?	□	□
3) wymienić elementy składowe typowej linii w asemblerze?	□	□
4) napisać program wpisujący liczby do rejestrów i narysować do niego algorytm?	□	□
5) napisać program zmieniający wartości bitów w poszczególnych rejestrach i narysować do niego algorytm?	□	□
6) napisać program wykorzystujący adresowanie pośrednie?	□	□
7) napisać program wykorzystujący adresowanie rejestrowe?	□	□
8) napisać program wykorzystujący pętle programową?	□	□
9) napisać program pracujący w pętli warunkowej?	□	□
10) napisać program zmieniający stany linii w portach?	□	□

4.4. Asembler mikrokontrolera 8051. Operacje arytmetyczno-logiczne. Skoki warunkowe

4.4.1. Materiał nauczania

Operacje arytmetyczne

Operacje arytmetyczne takie jak dodaj **ADD,** odejmij **SUBB**, mnóż **MUL**, dziel **DIV**, poprawka dziesiętna **DA** odbywają się tylko w powiązaniu z akumulatorem. Wynik zawsze zapisywany jest w akumulatorze. Wynik operacji **MUL** i **DIV** może być zarówno 8-bitowy jak i 16-bitowy, przy czym przy 16-bitowy wynik operacji mnóż jest zapisany w taki sposób, że bardziej znacząca cześć wyniku jest zapisana w **A,** a mniej znacząca w rejestrze **RP (B)**. Wynik 16- bitowy operacji **DIV** zapisany jest w taki sposób, że wynik całkowity dzielenia zapisany jest w **A**, natomiast reszta w rejestrze **B.** Operacje **MUL** i **DIV** wykonywane są na danych zapisanych w **A** i **B**.

Natomiast operacje zwiększ o jeden **INC** i zmniejsz o jeden **DEC** mogą być wykonywane na dowolnych rejestrach lub komórkach pamięci.

;Program sumujący 6 liczb.

W programie tym zapoznasz się, w jaki sposób można wykonać operacje dodawania 6 liczb, z czego dwie z nich należy zapisać w komórkach pamięci o adresie 43H i 54H, dwie w rejestrach roboczych R2 i R3 i dwie dodamy bezpośrednio. Liczby te zapisane szesnastkowo to: 36, 1A, 2B, 1F, 34 i 0B.

```
LJMP    START
        ORG     100H
START:
        MOV     43H,#36H        ;wpisz liczbę 36H do komórki o adresie 43H
        MOV     54H,#1AH        ;wpisz liczbę 1AH do komórki o adresie 54H
        MOV     R2,#2BH         ;wpisz liczbę 2BH do rejestru R2
        MOV     R3,#1FH         ;wpisz liczbę 1FH do rejestru R3
        MOV     A,43H           ;wpisz do akumulatora liczbę z komórki o adresie 43H
        ADD     A,54H           ;dodaj do akumulatora liczbę z komórki pamięci
                                 o adresie 54H
                                wynik w akumulatorze wynosi
                                A ← A+54H=36H+ 1AH=50H
        ADD     A,R2            ;dodaj do akumulatora liczbę z rejestru R2
                                wynik w akumulatorze wynosi
                                ;A ← A+R2=50H+ 2BH=7BH
        ADD     A,R3            ;dodaj do akumulatora liczbę z rejestru R3
                                wynik w akumulatorze wynosi
                                ;A ← A+R3=7BH+ 1FH=9AH
        ADD     A,#34H          ;dodaj do akumulatora liczbę 34H
                                wynik w akumulatorze wynosi
                                ;A←A+#34H=9AH+ 34H=CEH
        ADD     A,#0BH          ;dodaj do akumulatora liczbę 0BH
                                wynik w akumulatorze wynosi
                                A←A+#0BH=CEH+34H=D9H
        LJMP    $               ; pozostań w tej linii
```

W wyniku dodawania otrzymano liczbę D9H co odpowiada liczbie 217 dziesiętnie. Ponieważ liczba ta jest mniejsza od 255 czyli bit przeniesienia C w rejestrze flagowym nie uległ zmianie C=0, a cały wynik działania zapisany jest w akumulatorze.

W operacjach arytmetyczno-logicznych zdarza się, że wynik operacji jest większy niż 255. Jeżeli wynik operacji jest z zakresu 256-512 dziesiętnie to bit przeniesienia C w rejestrze flagowym przyjemnie wartość 1.

Program sumujący 2 liczby.

W programie tym zapoznasz się, w jaki sposób można wykonać operacje dodawania 2 liczb, których wynik przekracza liczbę 255.

```
LJMP    START
        ORG     100H
START:
        CLR     A               ;zeruj akumulator
        ADD     A,#128          ;dodaj do akumulatora liczbę 128
                                wynik w akumulatorze wynosi A←A+128=128
        ADD     A,#201          ;dodaj do akumulatora liczbę 201
                                wynik w akumulatorze wynosi
                                A←A+201=128+201=73=49H
                                wynik w akumulatorze to 49H,
                                zmieniona wartość bitu C=1
        LJMP    $               ; pozostań w tej linii
```

W programie tym nastąpiło przeniesienie, co oznacza ze wynik operacji przekroczył możliwą wielkość, jaka może być wpisana w akumulator i ustawiona została wartość bitu przeniesienia rejestru flagowego C=1.

Program obliczający następujące wyrażenie f=a*b – 3(c/d).

W programie tym zapoznasz się, w jaki sposób można wykonać operacje mnożenia, dzielenie i odejmowania. Liczby a, b, c i d są dobrane w taki sposób, aby otrzymany wynik poszczególnych działań był ośmiobitowy, czyli a=04H, b=1AH, c=FFH, d=11H. Wynik operacji zapisz w komórce o adresie 43H.

```
LJMP    START
        ORG     100H
START:
        MOV     R0,#04H         ;wpisz liczbę 04H do rejestru R0
        MOV     R1,#1AH         ;wpisz liczbę 1AH do rejestru R1
        MOV     R2,#FFH         ;wpisz liczbę FFH do rejestru R2
        MOV     R3,#11H         ;wpisz liczbę 11H do rejestru R3
        MOV     B,R1            ;wpisz do rejestru B liczbę z rejestru R1
        MOV     A,R0            ;wpisz do akumulatora liczbę rejestru R2
        MUL     A,B             ;pomnóż A ← A*B= 04H*1AH=68H
        MOV     R0,A            ;wpisz do rejestru R0 wynik mnożenia
        MOV     B,R4            ;wpisz do rejestru B liczbę z rejestru R4
        MOV     A,R3            ;wpisz do akumulatora liczbę z rejestru R4
        DIV     A,B             ; podziel A ← A/B= FFH/11H=0FH
        MOV     R1,A            ; wpisz do rejestru R1 wynik dzielenia
        MOV     A,R0            ;wpisz do akumulatora wynik mnożenia
        SUBB    A,R1            ;odejmij od wyniku mnożenia wynik dzielenia
                                ;A ← A - R1 - C= 68H-0FH - 0=59H
```

```
        SUBB    A,R1            ;odejmij od akumulatora wynik dzielenia
                                ;A ← A - R1- C = 59H-0FH - 0=4AH
        SUBB    A,R1            ;odejmij od akumulatora wynik dzielenia
                                ;A ← A - R1- C = 4AH-0FH - 0=36H, A = a*b – 3(c/d)
        MOV     43H,A           ;wyślij wynik działania do komórki pamięci o adresie 43H
        LJMP    $               ; pozostań w tej linii
```

W programie tym zastosowany rozkaz odejmowania SUBB odejmuje od wyniku wartość bitu pożyczki C w rejestrze flagowym.

Program zamieniający liczbę szesnastkową na liczbę dziesiętną w kodzie BCD.

W programie tym zapoznasz się, w jaki sposób zamienić liczbę szesnastkowa na liczbę dziesiętna. Program ten działa tylko przy założeniu, że liczba jest z zakresu od 0 do 99 dziesiętnie.

```
        LJMP    START
        ORG     100H
START:
        MOV     A,#3FH          ;wpisz liczbę 3FH do akumulatora
        MOV     B,#0AH          ;wpisz 0AH do rejestru B
        DIV     AB              ;podziel A/B, wyniku dzielenia liczba dziesiątek w A=6,
                                ;liczba jedności (reszta) w B=3
        SWAP    A               ;zamienia półbajty akumulatora między sobą
                                ; A7…A4 ↔ A3…A0
                                ;w akumulatorze znajduje się teraz liczba A=60H
        ADD     A,B             ;dodaj do akumulatora liczbę jednostek z B=3
                                ;wynik w akumulatorze A=63
        LJMP    $               ; pozostań w tej linii
```

Zastosowano tutaj rozkaz SWAP w tym celu, aby w akumulatorze znalazła się liczba dziesiątek na bitach $A_7…A_4$, ponieważ po dzieleniu była ona na bitach $A_3…A_0$. operacja dodawania wartości z rejestru B zmieniła tylko wartości bitów w $A_3…A_0$.

Operacje logiczne

W operacjach logicznych mnożenia logicznego **ANL** (ANL r,s), sumy logicznej **ORL** (ORL r,s), suma modulo 2 **XRL** (XRL r,s) wynik zapisywany jest w miejsce, z którego został pobrany argument r. Operacje logiczne wykonywane są zarówno na bitach jak i bajtach. Rozkaz negacji wartości **CPL** wykonywany jest tylko na akumulatorze. Operacje logiczne nie zmieniają wartości bitów w rejestrze flagowym

Program obliczający wartość wyrażenia logicznego $f(a,b,c,d,e) = (\overline{a}+b)(c+\overline{d})+aed$ **.**

W programie tym zapoznasz się, w jaki sposób obliczyć wartość wyrażenia logicznego argumenty w zadaniu w postaci szesnastkowej są następujące: a = 1FH, b= 2AH, c= A1H, d=73H e= 2BH.

```
        LJMP    START
        ORG     100H
START:
        MOV     R0,#1FH         ;wpisz liczbę 1FH do rejestru R0
        MOV     R1,#2AH         ;wpisz liczbę 2AH do rejestru R1
        MOV     R2,#A1H         ;wpisz liczbę A1H do rejestru R2
        MOV     R3,#73H         ;wpisz liczbę 73H do rejestru R3
        MOV     R4,#2BH         ;wpisz liczbę 2BH do rejestru R4
```

```
        MOV   A,R0          ;wpisz daną z rejestr R0 do rejestru A
        CPL   A             ;neguj A, A ← Ā = 1F̄ = E0
        ORL   R1,A          ;oblicz sumę logiczną,
                            R1← R1 OR A= 2AH + E0H=EAH
        MOV   A,R3          ;wpisz daną z rejestr R3 do rejestru A
        CPL   A             ;neguj A, A ← Ā = 73̄ = 8C
        ORL   A,R2          ;oblicz sumę logiczną,
                            A← A OR R2= 8CH + A1H =ADH
        ANL   R1,A          ;oblicz iloczyn logiczny,
                            R1←R1 AND A = EAH * ADH =A8H
        MOV   A,R0          ;wpisz daną z rejestr R0 do rejestru A
        ANL   A,R4          ;oblicz iloczyn logiczny,
                            A← A AND R4 = 1FH * 2BH =0BH
        ANL   A,R3          ;oblicz iloczyn logiczny,
                            A← A AND R3 = 0BH * 73H =03H
        ORL   A,R2          ;oblicz sumę logiczną,
                            A← A OR R2= 8CH + A1H =ADH
        ORL   A,R1          ;oblicz sumę logiczną,
                            A← A OR R1= ADH + EAH =EFH
        MOV   43H,A         ;wpisz wynik do komórki o adresie 43H
        LJMP  $             ; pozostań w tej linii
```

W programie tym w komentarzach pokazano stan, w jakim znajdują się poszczególne rejestry w czasie wykonania programu. Analiza programu i komentarzy pozwoli na zrozumienie, w jaki sposób przy pomocy mikrokontrolera można obliczyć wartość wyrażenia logicznego.

;Program obliczający wartość wyrażenia logicznego F= P>R AND (S=T OR U<V)**.**

W programie tym zapoznasz się, w jaki sposób obliczyć wartość wyrażenia logicznego. Jeżeli wyrażenie jest prawdziwe (TRUE) to należy wpisać do komórki pamięci 40H liczbę FFH, jeżeli fałszywe (FALSE) to należy wpisać 00H do komórki o adresie 41H. Symbole P, R, S, T, U i V są dowolnymi liczbami całkowitymi 8-bitowymi, które należy wpisać do komórek pamięci z zakresu 42H – 47H. W celu kompilacji programu do mikrokontrolera symbole liczb należy zastąpić konkretnymi wartościami.

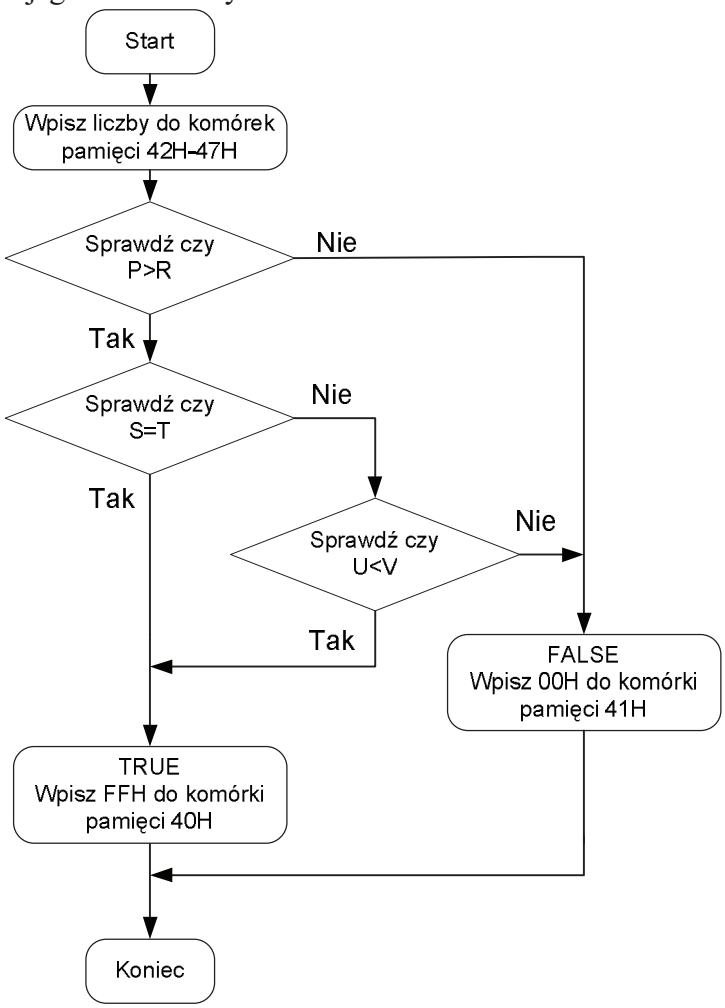
W programie tym w celu otrzymania wyniku TRUE wyrażenie P>R musi być spełnione i musi być spełniany jeden z warunków S=T lub U<V.

```
        LJMP  START
        ORG   100H
START:
        MOV   42H, #PH      ;wpisz liczbę #PH do komórki pamięci o adresie 42H
        MOV   43H, #RH      ;wpisz liczbę #RH do komórki pamięci o adresie 43H
        MOV   44H, #SH      ;wpisz liczbę #SH do komórki pamięci o adresie 44H
        MOV   45H, #TH      ;wpisz liczbę #TH do komórki pamięci o adresie 45H
        MOV   46H, #UH      ;wpisz liczbę #UH do komórki pamięci o adresie 46H
        MOV   47H, #VH      ;wpisz liczbę #VH do komórki pamięci o adresie 47H
        MOV   A,42H         ;wpisz liczbę P do akumulatora, A←P
        SUBB  A,43H         ;odejmij R od P, A←P-R-C
        JC    FALSE         ;skocz do etykiety FALSE jeżeli nastąpiła
                            pożyczka C=1 P<R
        MOV   A,44H         ;wpisz liczbę S do akumulatora, A←S
```

```
        SUBB    A,45H           ;odejmij T od S, A←S-T-C
        JZ      TRUE            ;skocz do etykiety TRUE jeżeli A=0
        CLR     C               ;zeruj bit C rejestru flagowego C←0
        MOV     A,46H           ;wpisz liczbę U do akumulatora, A←U
        SUBB    A,47H           ;odejmij V od U, A←U-V-C
        JNC     FALSE           ;skocz do etykiety FALSE jeżeli nie ma
                                ; pożyczki C=0 U>V
TRUE
        MOV     40H, #FFH       ;wpisz liczbę #FFH do komórki pamięci o adresie 40H
        LJMP    $               ; pozostań w tej linii
FALSE
        MOV     41H, #00H       ;wpisz liczbę #00H do komórki pamięci o adresie 41H
        LJMP    $               ; pozostań w tej linii
```

W programie tym wykorzystano wartości, jakie posiada bit C w rejestrze flagowy. Skoki JC, JZ i JNC oznaczają, że jeżeli następują to mikrokontroler przechodzi do wykonywania wywołanej etykiety pomijając linie programu miedzy etykietą a linią programu, w której skok jest zapisany. W przypadku gdy nie następuje skok mikrokontroler wykonuje następną linie programu .

W przypadku tego typu programu należy przed przystąpieniem do jego pisania narysować algorytm jego działania rys.14.



Rys.14. Algorytm programu obliczającego wartość wyrażenia logicznego F= P>R AND (S=T OR U<V)

4.4.2. Pytania sprawdzające

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

1. Które operacje arytmetyczne związane są z akumulatorem?
2. Jakie operacje arytmetyczne mogą posiadać wynik 16-bitowy?
3. Gdzie i w jaki sposób zapisany jest wynik operacji MUL?
4. Gdzie i w jaki sposób zapisany jest wynik operacji DIV?
5. Które z operacji arytmetycznych mogą być wykonywane na komórkach i rejestrach pamięci?
6. Co dzieje się, gdy wynik operacji arytmetycznej przekracza wartość 255?
7. W jaki sposób zmienia wartość bit C w rejestrze flagowym przy dodawaniu?
8. W jaki sposób zmienia wartość bit C w rejestrze flagowym przy odejmowaniu?
9. Co realizuje rozkaz SWAP?
10. Gdzie zapisywany jest wynik po wykonaniu operacji logicznej?
11. Czy operacje logiczne zmieniają wartości bitów w rejestrze flagowym?
12. W jaki sposób można zanegować wartość rejestru lub komórki pamięci?
13. Czy rozkazy logiczne związane są tylko z akumulatorem?
14. Na jakiej zasadzie działają rozkazy skoków warunkowych JZ, JNC i JC?
15. W jaki sposób wywoływane są pod programy (etykiety) przy wykonywaniu skoków warunkowych?

4.4.3. Ćwiczenia

Ćwiczenie 1

Celem ćwiczenia jest zapoznanie się z pisaniem programów arytmetycznych.
Na podstawie programów przedstawionych w materiale do nauczenia napisz program obliczający wartość następującej funkcji f=3x-2y+4z. Liczby x, y i z są dowolnymi liczbami 8-bitowymi, które należy zapisać w komórkach pamięci bądź rejestrach.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:

1) zapoznać się z programami przedstawionymi w materiale do nauczania,
2) zapoznać się z listą rozkazów mikrokontrolera 8051,
3) napisać program wraz z komentarzami,
4) ocenić poprawność wykonywanego ćwiczenia.

Wyposażenie stanowiska pracy:

- karty katalogowe układu 8051,
- lista rozkazów mikrokontrolera 8051,
- komputer,
- poradnik dla ucznia,
- dydaktyczny system mikroprocesorowy.

Ćwiczenie 2

Celem ćwiczenia jest zapoznanie z programowaniem mikrokontrolera 8051. Napisz program realizujący następującą funkcje f=2(x*y) – 2(z/y) + x. Liczby x, y i z są dowolnymi liczbami 8-bitowymi, które należy zapisać w komórkach pamięci bądź rejestrach.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zapoznać się z programami przedstawionymi w materiale do nauczania,
2) zapoznać się z listą rozkazów mikrokontrolera 8051,
3) napisać program,
4) ocenić poprawność wykonywanego ćwiczenia.

Wyposażenie stanowiska pracy:
– karty katalogowe układu 8051,
– lista rozkazów mikrokontrolera 8051,
– komputer,
– poradnik dla ucznia,
– dydaktyczny system mikroprocesorowy.

Ćwiczenie 3

Celem ćwiczenia jest zapoznanie z programowaniem mikrokontrolera 8051. Napisz program realizujący zmieniający liczbę binarną z zakresu 0…99 dziesiętnie na liczbę dziesiętną.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zapoznać się z programami przedstawionymi w materiale do nauczania,
2) zapoznać się z listą rozkazów mikrokontrolera 8051,
3) napisać program,
4) ocenić poprawność wykonywanego ćwiczenia.

Wyposażenie stanowiska pracy:
– karty katalogowe układu 8051,
– lista rozkazów mikrokontrolera 8051,
– komputer,
– poradnik dla ucznia,
– dydaktyczny system mikroprocesorowy.

Ćwiczenie 4

Celem ćwiczenia jest zapoznanie z programowaniem mikrokontrolera 8051. Napisz program realizujący następującą funkcje logiczną $f(a,b,c,d,e) = \overline{a}b + c\overline{d} + (a + \overline{c})(\overline{b} + d)$. Liczby a, b, c, d i e są dowolnymi liczbami 8-bitowymi, które należy zapisać w komórkach pamięci bądź rejestrach

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zapoznać się z programami przedstawionymi w materiale do nauczania,
2) zapoznać się z listą rozkazów mikrokontrolera 8051,
3) napisać program,
4) ocenić poprawność wykonywanego ćwiczenia.

Wyposażenie stanowiska pracy:
- karty katalogowe układu 8051,
- lista rozkazów mikrokontrolera 8051,
- komputer,
- poradnik dla ucznia,
- dydaktyczny system mikroprocesorowy.

Ćwiczenie 5

Celem ćwiczenia jest zapoznanie z programowaniem mikrokontrolera 8051. Napisz program obliczający następującą funkcje logiczną F= (P=R) OR (S>T AND U<V).. Jeżeli wyrażenie jest prawdziwe (TRUE) to należy wpisać do komórki pamięci 40H liczbę FFH, jeżeli fałszywe (FALSE) to należy wpisać 00H do komórki o adresie 40H. Symbole P, R, S, T, U i V są dowolnymi liczbami całkowitymi 8-bitowymi, które należy wpisać do komórek pamięci z zakresu 42H – 47H.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zapoznać się z programami przedstawionymi w materiale do nauczania,
2) zapoznać się z listą rozkazów mikrokontrolera 8051,
3) narysować algorytm działania programu,
4) napisać program,
5) ocenić poprawność wykonywanego ćwiczenia.

Wyposażenie stanowiska pracy:
- karty katalogowe układu 8051,
- lista rozkazów mikrokontrolera 8051,
- komputer,
- poradnik dla ucznia,
- dydaktyczny system mikroprocesorowy.

4.4.3 Sprawdzian postępów

Czy potrafisz:	**Tak**	**Nie**
1) napisać program sumujący liczby?	□	□
2) napisać program obliczający wartość określonego wyrażenia arytmetycznego uwzględniający rejestr flagowy?	□	□
3) napisać program zmieniający liczbę szesnastkową i binarną na dziesiętną?	□	□
4) napisać program obliczający wyrażenie logiczne?	□	□
5) narysować algorytm do programu wykorzystującego skoki warunkowe?	□	□
6) napisać program w którym poprzez skoki warunkowe wywoływane są podprogramy?	□	□
7) napisać program sprawdzający prawdziwości wyrażenia logicznego?	□	□
8) skorzystać z listy rozkazów mikrokontrolera 8051?	□	□
9) uruchomić program a dydaktycznym systemie mikroprocesorowym?	□	□
10) sprawdzić poprawność działania programu?	□	□

6. LITERATURA

1. Dyrycz K.P., Kowalski Cz.T., Żarczyński Z.: Podstawy techniki mikroprocesorowe. Oficyna wyd. Politechniki Wrocławskiej, Wrocław 1999
2. Gałka P., Gałka P.: Podstawy programowania mikrokontrolera 8051. Wyd. ZNI „Nikom", Warszawa 1995
3. Małysiak H., Pochopień B., Podsiadło P., Wróbel E.: Modułowe systemy mikrokomputerowe. WNT, Warszawa 1990
4. Pełka R.: Mikrokontrolery architektura programowanie zastosowania. Wyd. Komunikacji i Łączności, Warszawa 1999
5. Pieńkos J., Moszczyński S., Pluta A.: Układy mikroprocesorowe 808/8085. Wyd. Komunikacji i Łączności, Warszawa 1988
6. Zieliński B.: Układy mikroprocesorowe. Przykłady rozwiązań. Wyd. Helion, Gliwice 2002
7. Noty katalogowe układów