ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1994
Seria: ELEKTRONIKA z. 3 Nr kol. 1254

Witold BARAN
Instytut Elektroniki
Politechnika Śląska

OPROGRAMOWANIE SYSTEMU WIELOPROCESOROWEGO

Streszczenie. W pracy opisano oprogramowanie systemu wieloprocesorowego zbudowanego na podstawie mikroprocesorów rodziny Intel 8085. Przedstawiono podstawowe elementy oprogramowania oraz informacje istotne dla uruchamiania zadań opierając się na zbudowanym systemie wieloprocesorowym.

The software of a multiprocessor system

Summary. The article presents the software of the multiprocessor system based on the Intel 8085 microprocessors. The basic elements of the software are presented as well as the information which is necessary for troubleshooting of the constructed multiprocessor system.

Математическое обеспечение мультипроцессорных систем

Резюме. Статья предлагает математическое обеспечение мультипроцессорной системы основанной на микропроцессорах Интел 8085. Представлены основные слагаемые математического обеспечения, а также сведения, необходимые для отлатки задач с применением построенной мультипроцессорной системы.

1. WSTĘP

Oprogramowanie systemu wieloprocesorowego stanowi jedną z najtrudniejszych dziedzin współczesnego programowania. Stworzenie systemu pozwalającego wykorzystać w pełni możliwości pracy równoległej takiego systemu jest czasochłonne, a stąd kosztowne. Należy również podkreślić, że celem tej pracy nie było stworzenie oprogramowania systemu pracującego równolegle, lecz sprawdzenie pewnej koncepcji badawczej, a następnie zbudowanie i uruchomienie modelu systemu wieloprocesorowego służącego jako baza do jednego z przyszłych stanowisk laboratoryjnych w Laboratorium Systemów Wielomikroprocesorowych. Z tych założeń wynikają przyjęte uproszczenia w zakresie oprogramowania oraz przeniesienie części wysiłku związanego z rozwiązaniem problemu rozdziału zadań pomiędzy poszczególne mikroprocesory na twórców przyszłych aplikacji. Wynika stąd także dydaktyczny aspekt tej sprawy - możliwość pokazania, na przykładach, wpływu podziału zadań na szybkość działania całego systemu wieloprocesorowego.

Kolejnym problemem, jaki stanął przed twórcami systemu, było rozwiązanie kwestii, czy wyposażyć system wieloprocesorowy w środki umożliwiające przygotowywanie kolejnych zadań, lub też ograniczyć się do budowy prostego interfejsu pozwalającego transmitować zadania, dane i wyniki do systemu nadzorczego. Ze względu na koszt i prostotę wybrano to drugie rozwiązanie.

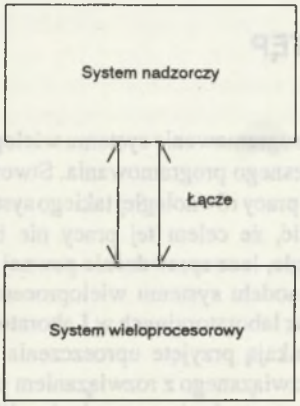
Reasumując założono, że oprogramowanie systemu wieloprocesorowego powinno być możliwie proste, lecz uniwersalne. Podział zadań w ramach systemu przekazano na barki przyszłych użytkowników. Zadania do wykonania oraz dane będą przygotowywane w zewnętrznym systemie nadzorczym, przy użyciu ogólnie dostępnych środków programistycznych oraz oprogramowania pozwalającego zawiadować systemem wieloprocesorowym z poziomu systemu nadzorczego.

2. WYBÓR RODZAJU ŁĄCZA SYSTEMU NADZORCZEGO I SYSTEMU WIELOPROCESOROWEGO

Jak już wcześniej wspomniano, przyjęta została koncepcja swoistego "zdalnego sterowania" systemu wieloprocesorowego przez system nadzorczy. Pokazuje to rys. 1.

Jako system nadzorczy został wybrany komputer klasy IBM PC, natomiast łącze zrealizowano przy użyciu łącza szeregowego RS232C. Decyzja o użyciu tego typu łącza wynikała z jego dużej odporności na zakłócenia, przy długości połączenia ok. 3 m. Możliwość wykorzystania w urządzeniu łącza szeregowego, dużo wolniejszego w stosunku do równoległego, wynikała z faktu projektowania użycia systemu wieloprocesorowego do obliczeń numerycznych, co wiąże się ze stosunkowo małą częstotliwością wymiany informacji między systemami. Przy obliczeniach, przeważnie, czas przesyłania danych jest dużo mniejszy niż czas wykonywania obliczeń. Widać tu oczywiście kolejną trudność wyłaniającą się przed użytkownikiem realizującym obliczenia. Musi on nie tylko dobrze opracować rozdział zadań w systemie wieloprocesorowym, ale także ustalić strategię przesyłania danych podczas obliczeń, np. przez odpowiednią kolejność ich wykonywania. Przyjęty typ łącza ma jeszcze tę zaletę, że nie wymaga żadnych zmian sprzętowych w systemie nadzorczym IBM PC. Na wyposażeniu tego komputera znajduje się złącze RS232C. Po wykonanych doświadczeniach przyjęto następujące parametry transmisji szeregowej:

- szybkość 4800 bodów,
- 8 bitów danych,
- bit parzystości
- 1 bit stopu.



Rys. 1 Ogólna zasada współpracy systemu nadzorczego z systemem wieloprocesorowym
Fig. 1 General rule of the co-operation between the host and multiprocessor system

3. WYMIANA INFORMACJI W SYSTEMIE WIELOPROCESOROWYM

3.1. Zasady współpracy systemu nadzorczego z systemem wieloprocesorowym

Przyjęcie założenia, że zadania i dane są przygotowywane w systemie nadzorczym, spowodowało, że mogły zostać użyte standardowe środki do tworzenia oprogramowania na mikroprocesorze 8085, gdyż taki właśnie typ procesora wykorzystano w systemie wieloprocesorowym. Użycie innego typu procesora w systemie wieloprocesorowym niż w systemie nadzorczym pociągnęło za sobą także konieczność stosowania kompilatorów i asemblerów generujących kod wynikowy dla tego procesora. Istnieje oczywiście możliwość użycia programów typu crossasembler, pracujących pod kontrolą systemu PC lub MS DOS, a generujących kod wynikowy dla mikroprocesora 8085, jednak przyjęto inne rozwiązanie. Użyty został program symulujący otoczenie programowe systemu CP/M w systemie PC lub MS DOS. Dzięki takiemu rozwiązaniu dostępne stało się bardzo bogate oprogramowanie narzędziowe generujące kod wynikowy dla procesorów rodziny MCS80 i MSC85 firmy Intel, podstawowych dla systemu CP/M. Jako podstawowy język do tworzenia aplikacji w systemie, ze względu na możliwość maksymalnej optymalizacji obliczeń, przyjęto język asembler 8085. Wynika stąd następująca metodologia postępowania przy tworzeniu oprogramowania użytkowego dla systemu wieloprocesorowego. Program źródłowy dla asemblera przygotowywany jest przy użyciu dowolnego edytora tekstów, pracującego w trybie nie-dokument. Następnie przy użyciu symulatora i standardowych asemblerów i linkerów systemu CP/M (ASM,LINK,M80,L80) generowany jest program wynikowy, który może być wykonywany w procesorze MASTER systemu wieloprocesorowego. Podobnie przygotowywane są zbiory danych. Ponieważ żaden program nie jest wolny od błędów, uruchomienie programu może również nastąpić przy użyciu symulatora CP/M i programów uruchomieniowych np. DDT lub SID. Tak przygotowane zbiory będą następnie transmitowane do systemu wieloprocesorowego i tam wykonane.
Oprogramowanie opracowane w ramach tej pracy podzielić można na dwa rodzaje :

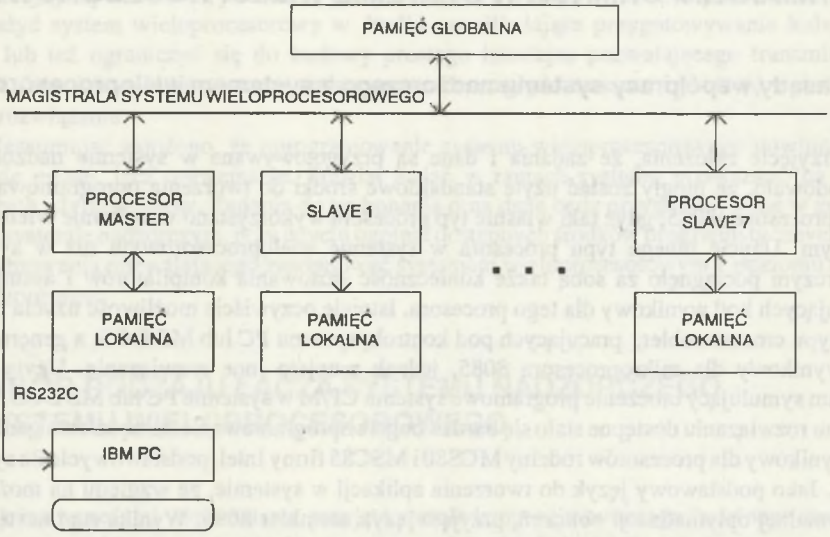
- oprogramowanie systemu wieloprocesorowego,
- oprogramowanie systemu nadzorczego.

Zadaniem pierwszego jest realizacja konkretnych zadań obliczeniowych, wstępne testowanie sprzętu (procesory SLAVE i MASTER). Procesor MASTER realizuje także komunikację z systemem nadzorczym i możliwości ładowania zadania i danych oraz odczyt wyników. Programy te zawarte są w pamięciach stałych poszczególnych pakietów procesorów.

Zadaniem oprogramowania systemu nadzorczego jest stworzenie środowiska pozwalającego na uruchamianie wcześniej napisanych aplikacji w systemie wieloprocesorowym.

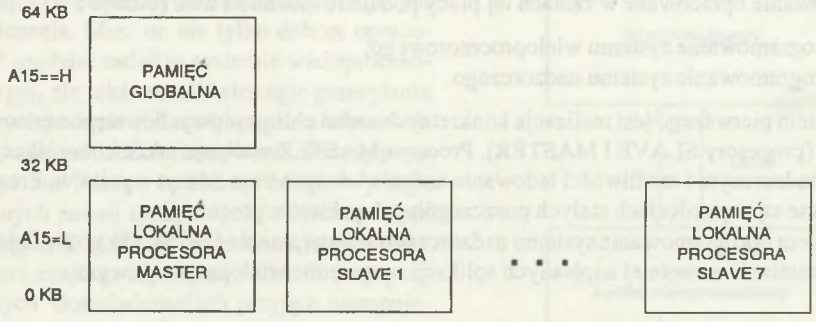
3.2. Zasady współpracy procesora MASTER z procesorami SLAVE

Z rys. 2. wynika, że procesor MASTER może komunikować się z procesorami SLAVE poprzez pamięć globalną. W omawianym systemie zastosowane zostały procesory 8-bitowe 8085 adresujące jedynie do 64 KB pamięci. Rozdzielenie adresów pamięci globalnej i lokalnej



Rys. 2. Schemat ilustrujący zasady wymiany informacji w systemie wieloprocesorowym
Fig. 2. The scheme illustrating rules of the information interchange in the multiprocessor system

osiągniete zostało przez dekodowanie stanu najstarszego bitu adresu (A15) każdego procesora. Dla A15=L dekodowana jest pamięć lokalna, natomiast dla A15=H ma miejsce dostęp do pamięci globalnej. Zasadę tę ilustruje rys. 3. Dokładniejsze omówienie sposobu wykorzystania pamięci globalnej do komunikacji MASTER-SLAVE znajduje się w punkcie 5.4.2.



Rys. 3. Podział pamięci procesora MASTER i procesorów SLAVE
Fig. 3. Sharing memory of the MASTER processor and the SLAVE processors

4. OPIS PROGRAMU STERUJĄCEGO MASTER MULTI

4.1. Miejsce programu Master-Multi w systemie wieloprocesorowym

Program Master-Multi stanowi podstawowe środowisko pracy dla użytkownika systemu wieloprocesorowego. Stanowi on nadrzędny program w całym systemie, jest łącznikiem między systemem sterującym - IBM PC i systemem wieloprocesorowym. Program Master- Multi działa pod nadzorem systemu operacyjnego MS DOS lub PC DOS wersji 3.30 lub wyższej.

4.2. Podstawowe funkcje programu Master-Multi

Z umiejscowienia programu w systemie wynikają jego następujące funkcje:

1. komunikacja z użytkownikiem w otoczeniu systemu operacyjnego komputera nadzorczego;
2. komunikacja z procesorem Master i procesorami Slave poprzez łącze szeregowe;
3. operowanie zbiorami z wykorzystaniem zasobów komputera nadzorczego.

4.3. Uproszczony schemat blokowy programu

Pierwszą czynnością programu Master-Multi jest inicjalizacja kanału COM oraz zmiennych programu. Następnie program obsługuje menu główne programu. Po wyborze jednej z opcji wykonywana jest obsługa tej opcji, po jej zakończeniu następuje powrót do menu głównego. Z menu głównego możliwy jest także powrót do systemu operacyjnego komputera nadzorczego. Ogólny schemat blokowy programu przedstawia rys. 4.

4.4. Opis programu Master-Multi

Szczegółowy opis zasad współpracy programów Master-Multi i Master będzie zamieszczony w punkcie 5.4.1. W tym punkcie główny nacisk został położony na opisanie programu Master-Multi od strony użytkownika i jego współpracy z systemem MS DOS.

Program sterujący całym systemem wieloprocesorowym został napisany z zastosowaniem techniki rozwijanego menu okienkowego. Każde z okien odpowiada jednej funkcji lub ich grupie. Po poprawnej inicjalizacji programu, brak sygnalizacji błędów, mamy do dyspozycji menu główne, a w nim cztery opcje. Są to:

File
Transmit
Run
Options.

Należy tu zwrócić uwagę na fakt, że wszystkie komunikaty i napisy pojawiające się na ekranie monitora są w języku angielskim. Fakt ten nie wynika ze snobizmu autora programu, lecz z niechęci do kaleczenia naszego języka ojczystego - brak polskich liter, a nawet jednolitego standardu ich stosowania.

Zasadą przyjętą przez autora jest, że wyboru danej opcji dokonać można przez naciśnięcie pierwszej litery (małej lub dużej) słowa opisującego opcję lub przez naprowadzenie podświetlonego prostokąta na właściwą opcję i naciśnięcie klawisza Enter. Wycofanie się z realizacji danej opcji następuje przez naciśnięcie klawisza Esc. Akceptacja opcji lub parametrów ma miejsce przez naciśnięcie Enter. W przypadku sytuacji wyboru jako domniemany dany jest przypadek dający ewentualne najmniejsze szkody (jest on wyświetlany jako podświetlony lub w nawiasach kwadratowych "[ ]"). Zakończenie wykonywania programu z poziomu menu głównego następuje po naciśnięciu kombinacji klawiszy Alt i X. Program zawsze proponuje standardowe rozszerzenia zbiorów:

cfg - zbiór z konfiguracją;
dat - zbiór z danymi;
pro - zbiór z programem;
res - zbiór z wynikami;

4.4.1. Opis opcji File

Opcja ta zapewnia obsługę zbiorów dyskowych w systemie operacyjmym MS DOS. Możliwe są tu następujące opcje:

Open
Save
Change dir
Dos shell
Exit

Opcja **Open** daje możliwość odczytu zbioru z danymi lub programem do bufora w pamięci komputera sterującego.Save pozwala zapamiętać w zbiorze dyskowym wyniki obliczeń przetransmitowane wcześniej do bufora w pamięci systemu nadzorczego.

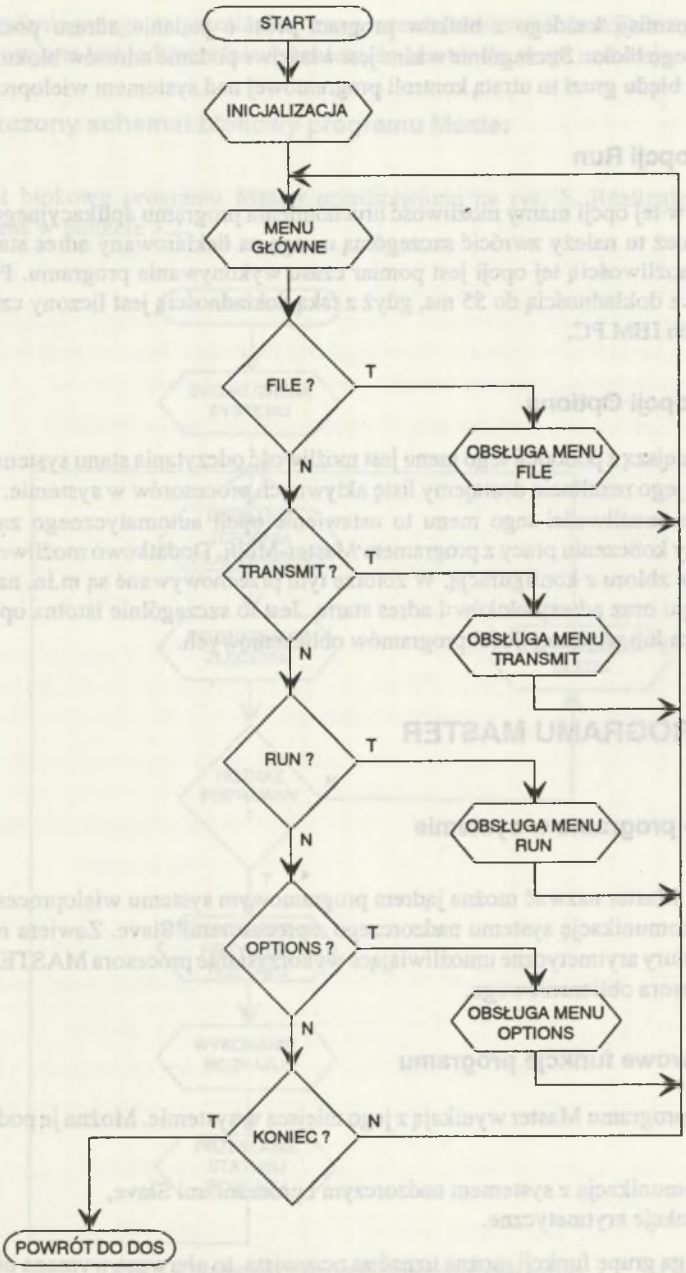
Opcja **Change** dir zapewnia możliwość zmiany ścieżki dostępu do zbiorów na dysku.

Opcja **Dos shell** umożliwia wykonanie operacji dosowskiej. Powrót do programu następuje po wydaniu polecenia **Exit.**

Opcja **Exit** umożliwia zakończenie wykonywania programu Master-Multi i powrót do systemu operacyjnego.

4.4.2. Opis opcji Transmit

Opcja ta pozwala na transmisję programu (Program) i danych (Data) do procesora Master oraz rezultatów (Results) z tego procesora. W transmisji wykorzystywane są dane zapamiętane w buforze pamięci komputera nadzorczego, ewentualnie dane są tam zapamiętywane.



Rys. 4. Ogólny schemat blokowy programu Master-Multi
Fig. 4. General block diagram of the Master-Multi programme

Przy transmisji każdego z bloków program prosi o padanie adresu początku i końca transmitowanego bloku. Szczególnie ważne jest właściwe podanie adresów bloku z programem. W przypadku błędu grozi to utratą kontroli programowej nad systemem wieloprocesorowym.

4.4.3. Opis opcji Run

Dopiero w tej opcji mamy możliwość uruchomienia programu aplikacyjnego od zadanego adresu. Również tu należy zwrócić szczególną uwagę na deklarowany adres startu programu. Dodatkową możliwością tej opcji jest pomiar czasu wykonywania programu. Pomiar ten jest wykonywany z dokładnością do 55 ms, gdyż z taką dokładnością jest liczony czas systemowy w komputerach IBM PC.

4.4.4. Opis opcji Options

Najważniejszą z podopcji tego menu jest możliwość odczytania stanu systemu wieloprocesorowego. W jego rezultacie dostajemy listę aktywnych procesorów w systemie.

Pozostałe możliwości tego menu to ustawienie opcji automatycznego zapisu zbioru z wynikami przy kończeniu pracy z programem Master-Multi. Dodatkowo możliwe jest zapisanie lub odczytanie zbioru z konfiguracją. W zbiorze tym przechowywane są m.in. nazwy zbiorów, ścieżka dostępu oraz adresy bloków i adres startu. Jest to szczególnie istotna opcja przy kilku użytkownikach lub większej ilości programów obliczeniowych.

5. OPIS PROGRAMU MASTER

5.1. Miejsce programu w systemie

Program Master nazwać można jądrem programowym systemu wieloprocesorowego. Pozwala on na komunikację systemu nadzorczego z procesorami Slave. Zawiera również standardowe procedury arytmetyczne umożliwiające wykorzystanie procesora MASTER jako jeszcze jednego procesora obliczeniowego.

5.2. Podstawowe funkcje programu

Funkcje programu Master wynikają z jego miejsca w systemie. Można je podzielić na dwie grupy:

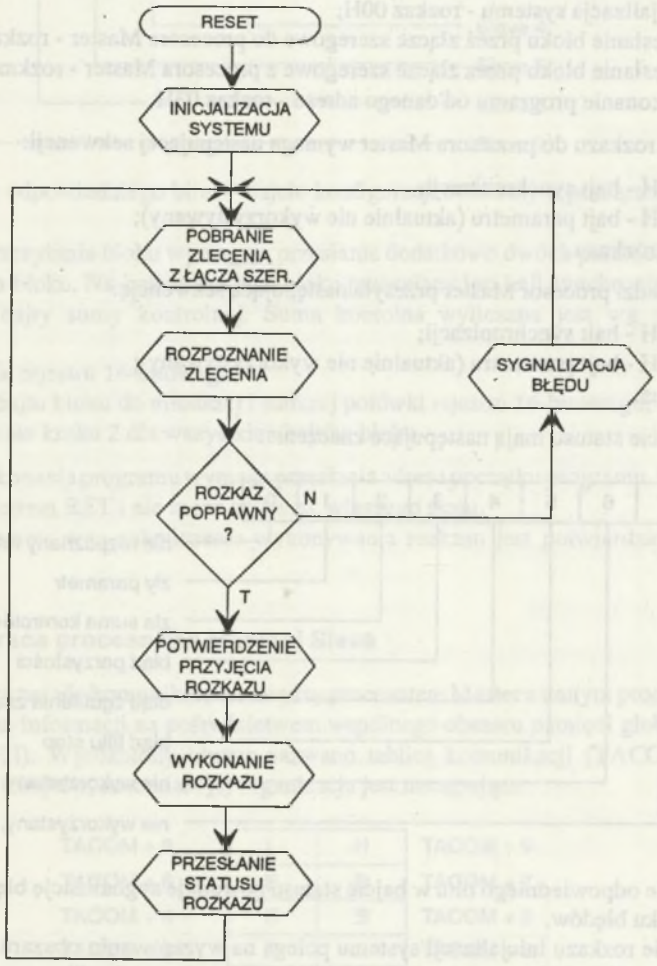
1. komunikacja z systemem nadzorczym i procesorami Slave,
2. funkcje arytmetyczne.

O ile drugą grupę funkcji można uznać za oczywistą, to pierwsza wymaga nieco komentarza. Po impulsie RESET zadaniem procesora Master jest zainicjowanie procesorów Slave, łącza szeregowego, oraz podstawowych zmiennych. Następnie procesor oczekuje na zlecenia z systemu nadzorczego. Po odebraniu zlecenia Master rozpoznaje je i wykonuje. Zakończenie

wykonania zadania polega na przesłaniu przez procesor Master informacji do systemu nadzorczego. Informacja ta jest jednocześnie sygnałem, że Master oczekuje na kolejne zadanie.

5.3. Uproszczony schemat blokowy programu Master

Schemat blokowy programu Master przedstawiono na rys. 5. Realizuje on wszystkie funkcje opisane w punkcie 5.2.



Rys. 5. Uproszczony schemat blokowy programu Master
Fig. 5. Simplified block diagram of the Master programme

5.4. Obsługa programu Master

Obsługa programu Master zostanie przedstawiona z punktu widzenia jego sterowania przez system nadzorczy oraz sterowania przez procesor Master procesorów Slave.

5.4.1. Sterowanie programem Master przez system nadzorczy

Sterowanie programem Master odbywa się za pośrednictwem następujących rozkazów:

1. inicjalizacja systemu - rozkaz 00H;
2. przesłanie bloku przez złącze szeregowe do procesora Master - rozkaz 01H;
3. przesłanie bloku przez złącze szeregowe z procesora Master - rozkaz 02H;
4. wykonanie programu od danego adresu - rozkaz 03H.

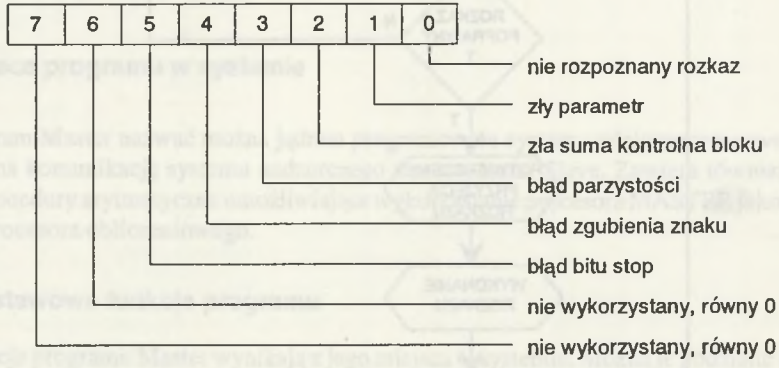
Przesłanie rozkazu do procesora Master wymaga następującej sekwencji:

1. 0E6H - bajt synchronizacji;
2. 0FFH - bajt parametru (aktualnie nie wykorzystywany);
3. bajt rozkazu.

W odpowiedzi procesor Master przesyła następującą sekwencję:

1. 0E6H - bajt synchronizacji;
2. 0FFH - bajt parametru (aktualnie nie wykorzystywany);
3. status.

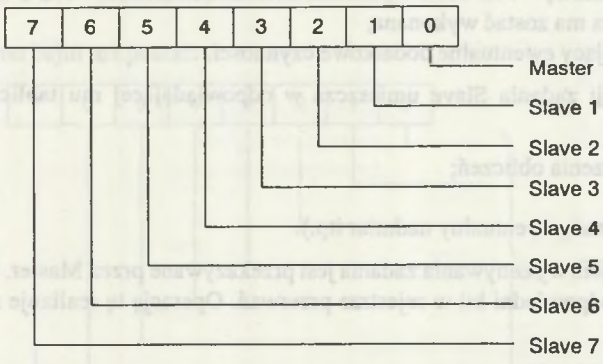
Bity w bajcie statusu mają następujące znaczenie:



Ustawienie odpowiedniego bitu w bajcie statusu powoduje sygnalizację błędu. Status 00H świadczy o braku błędów.

Wykonanie rozkazu inicjalizacji systemu polega na wyzerowaniu obszaru tablic parametrów procesorów Slave, a następnie stwierdzeniu, które z procesorów Slave są podłączone do systemu. Jako rezultat inicjalizacji Master przesyła bajt konfiguracji systemu oraz bajt zawierający liczbę procesorów w systemie.

Znaczenie bitów w bajcie konfiguracji jest następujące:



Ustawienie odpowiedniego bitu w bajcie konfiguracji świadczy o podłączeniu procesora do systemu.

Rozkazy przesyłania bloku wymagają przesłania dodatkowo dwóch parametrów, adresów początku i końca bloku. Na końcu każdego bloku przesyłany jest bajt synchronizacji, 0E6H, a następnie dwa bajty sumy kontrolnej. Suma kontolna wyliczana jest wg następującego algorytmu:

1. zerowanie rejestru 16-bitowego;
2. dodanie bajtu bloku do młodszej i starszej połówki rejestru 16-bitowego;
3. powtórzenie kroku 2 dla wszystkich bajtów bloku.

Rozkaz wykonania programu wymaga przesłania adresu początku programu. Program musi kończyć się rozkazem RET i nie może ustawiać własnego stosu.

Każdy parametr oraz zakończenie wykonywania rozkazu jest potwierdzane sekwencją statusu.

5.4.2. Współpraca procesorów Master i Slave

Podstawową zasadę komunikacji pomiędzy procesorem Master a danym procesorem Slave stanowi wymiana informacji za pośrednictwem wspólnego obszaru pamięci globalnej (adresy 8000H - 0FFFFH). Wyróżniony obszar nazwano tablicą komunikacji (TACOM). Długość tablicy wynosi 10 bajtów, natomiast jej organizacja jest następująca:

TACOM + 8	L	H	TACOM + 9
TACOM + 6	E	D	TACOM + 7
TACOM + 4	C	B	TACOM + 5
TACOM + 2	F	A	TACOM + 3
TACOM	PAR	FUN	TACOM + 1

Za pośrednictwem tablicy TACOM procesor Master przekazuje odpowiedniemu procesorowi Slave następujące dane:

1. adresy argumentów w pamięci globalnej;
2. długości argumentów;
3. kod operacji, jaka ma zostać wykonana;
4. parametr określający ewentualne dodatkowe czynności.

W trakcie realizacji zadania Slave umieszcza w odpowiadającej mu tablicy TACOM informacje, takie jak:

1. znacznik zakończenia obliczeń;
2. adres wyniku;
3. status wyniku (znak, ewentualny nadmiar itp.).

Polecenie rozpoczęcia wykonywania zadania jest przekazywane przez Master, który drogą programową wzbudza odpowiedni bit w rejestrze przerwań. Operację tę realizuje następująca sekwencja rozkazów:

```
MVI   A,MASKA
OUT   INTR
```

Ustawienie w rejestrze przerwań bitu i powoduje wysłanie sygnału przerwania INT_i do i-tego procesora. Jak widać, możliwe jest w ten sposób równoczesne pobudzenie wszystkich procesorów Slave w systemie. Po przyjęciu przerwania dany procesor podrzędny wysyła sygnał potwierdzenia, kasujący odpowiadające mu zgłoszenie w rejestrze przerwań. Rejestr umieszczony jest w przestrzeni adresowej globalnych urządzeń we/wy. Tak więc, oprócz modułu Master, każdy procesor Slave posiada możliwości programowej zmiany jego zawartości. Dany Slave mógłby więc kasować programowo odpowiadające mu zgłoszenie przerwania. Ponieważ jednak nie ma możliwości odczytu zawartości rejestru, niemożliwe jest wyzerowanie określonego bitu bez zmiany bitów pozostałych. Przyjęto więc zasadę, że programowy dostęp do rejestru przerwań wykorzystuje jedynie procesor Master podczas zlecania zadań, natomiast procesory Slave kasują określone bity rejestru sprzętowo.

Format tablicy TACOM

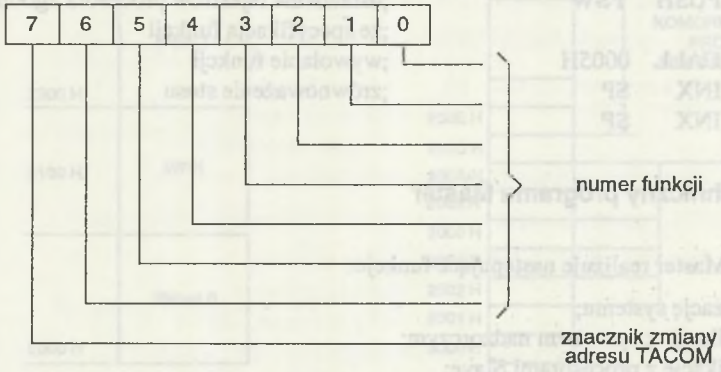
Poszczególne pola tablicy mają następujące znaczenie:

1. HL - adres pierwszego argumentu; ustalono, że w polu pierwszego argumentu umieszczany jest wynik; w sytuacji, gdy z pewnych względów nie spełnia tej reguły, Slave umieszcza w komórkach HL zmodyfikowany adres wyniku;
2. DE - adres drugiego argumentu;
3. BC - długość argumentów pomniejszona o 1;
4. AF - status wyniku;

Wynik	A	F
Zerowy	0	Z
Ujemny	1	M
Dodatni	2	P
Nadmiar	3	CY

5. PAR - definiuje czynności dodatkowe procesora Slave;
6. FUN - określa kod zadania zleconego Slave'owi

Format bajtu ma postać:



Kody funkcji wykonywanych przez procesory Slave:

0 - potwierdzenie sprawności procesora;
1 - dodawanie stałoprzecinkowe;
2 - odejmowanie stałoprzecinkowe;
3 - mnożenie stałoprzecinkowe;
4 - dzielenie stałoprzecinkowe;
5 - suma N liczb zmiennoprzecinkowych;
6 - pobranie i wykonanie programu;
7 - dodawanie zmiennoprzecinkowe;
8 - odejmowanie zmiennoprzecinkowe;
9 - mnożenie zmiennoprzecinkowe;
10 - dzielenie zmiennoprzecinkowe;
11 - mnożenie/dzielenie przez potęgę dwóch;
12 - porównanie zmiennoprzecinkowe specjalnego;
13 - obliczanie pierwiastka kwadratowego;
14 - przenoszenie pola;
15 - zerowanie pola;
16 - inkrementacja zawartości pola;
17 - przesunięcie w lewo z zerem;
18 - przesunięcie w prawo z zerem;
19 - przesunięcie w lewo z przeniesieniem;
20 - przesunięcie cykliczne w prawo;
21 - przesuwanie o cztery bity;
22 - dopełnienie do dwóch - zmiana znaku;

Program załadowany przez system nadzorczy może korzystać z procedur standardowych programu Master. Wywołanie funkcji zapewnia następująca sekwencja rozkazów:

```
MVI   A,KOD_FUNKCJI   ;kod funkcji przekazywany przez stos
PUSH  PSW             ;ustawienie rejestrów procesora zgodnie
                      ;ze specyfikacją funkcji
CALL  0005H           ;wywołanie funkcji
INX   SP              ;zrównoważenie stosu
INX   SP
```

5.5. Opis techniczny programu Master

Program Master realizuje następujące funkcje:

1. inicjalizację systemu;
2. komunikację z systemem nadzorczym;
3. komunikację z procesorami Slave;
4. działania arytmetyczne na liczbach wielokrotnej precyzji.

5.5.1. Organizacja pamięci

Procesor Master posiada 8 kB pamięci EPROM i 24 kB pamięci RAM. Do realizacji podstawowych funkcji programu Master wystarcza obszar 1 kB. Dlatego też obszar pamięci RAM podzielony został następująco (rys. 6):

- strona zerowa (256B)
- trzy obszary robocze procedur arytmetycznych o długości równej najdłuższym spodziewanym argumentom (3 x 256B).

Strona zerowa zawiera zmienne systemowe i komórki robocze procedur arytmetycznych. Pozostały obszar pamięci RAM może być wykorzystany do ładowania programów (patrz funkcja numer 5 programu Master).

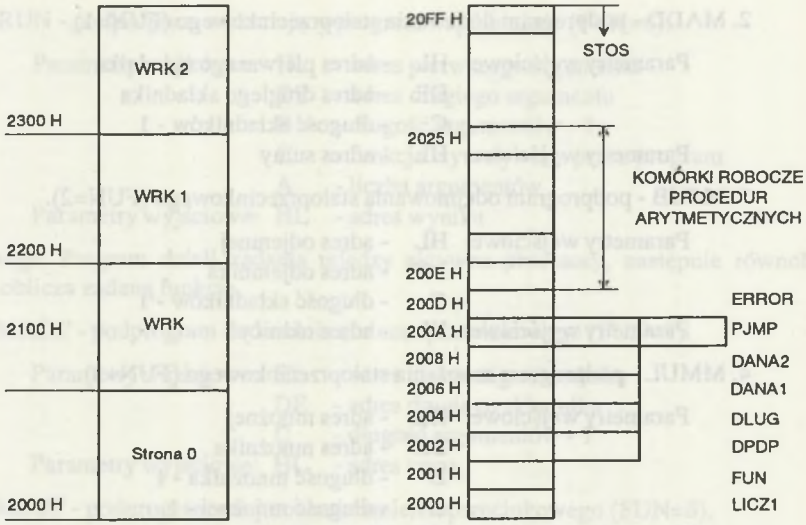
5.5.2. Inicjalizacja systemu

Po impulsie RESET ustawiany jest stos programu, programowane układy łącza szeregowego oraz inicjalizowane procesory Slave.

5.5.3. Komunikacja z systemem nadzorczym

Komunikacja przez łącze szeregowe realizowana jest w następującej sekwencji:

1. wyzerowanie bajtu statusu i opróżnienie bufora transmisji;
2. pobranie rozkazu z łącza szeregowego;



Rys. 6. Organizacja pamięci RAM procesora Master
Fig. 6. Organisation of the RAM memory of the Master processor

3. rozpoznanie rozkazu;
4. obliczenie skoku procedury realizującej rozkaz, adres procedury zapisany jest w tablicy INSTAB; rzeczywisty adres zapisywany jest do obszaru zmiennej PJMP;
5. przesłanie statusu do systemu nadzorczego - potwierdzenie odebrania rozkazu;
6. wykonanie rozkazu;
7. przesłanie statusu wykonania rozkazu.

5.5.4. Komunikacja z procesorami Slave

Komunikacja z procesorami Slave realizawana jest za pośrednictwem tablic parametrów TACOM. Program wczytywany przez łącze szeregowe może wywoływać procedury standardowe programu Master. Metodologia wywoływania tych funkcji została opisana w punkcie 5.4.2.

5.5.5. Opis podprogramów programu Master wywoływanych przez skok pod adres 0005H

1. INSLAV - podprogram inicjalizacji procesorów Slave (FUN=0).

Parametry wejściowe: - - -
Parametry wyjściowe: - - -

Uwagi: Zeruje obszar tablic TACOM, następnie inicjalizuje procesory Slave wywołując podrogram INZE.

2. MADD - podprogram dodawania stałoprzecinkowego (FUN=1).

Parametry wejściowe: HL - adres pierwszego składnika
DE - adres drugiego składnika
C - długość składników - 1
Parametry wyjściowe: HL - adres sumy

3. MSUB - podprogram odejmowania stałoprzecinkowego (FUN=2).

Parametry wejściowe: HL - adres odjemnej
DE - adres odjemnika
C - długość składników - 1
Parametry wyjściowe: HL - adres różnicy

4. MMUL - podprogram mnożenia stałoprzecinkowego (FUN=3).

Parametry wejściowe: HL - adres mnożnej
DE - adres mnożnika
B - długość mnożnika - 1
C - długość mnożnej - 1
Parametry wyjściowe: HL - adres iloczynu

Uwagi: Realizowane jest mnożenie dwóch liczb o różnej długości, nie przekraczającej 128 bajtów. Długość iloczynu może wynieść do 256 bajtów. W przypadku gdy wynik jest dłuższy od mnożnej, zajmuje on dodatkowe pole przed mnożną. Zmodyfikowany adres iloczynu umieszczany jest w polu HL. W przypadku nadmiaru mnożna zostaje zniszczona.

5. MDIV - podprogram dzielenia stałoprzecinkowego (FUN=4).

Parametry wejściowe: HL - adres dzielnej
DE - adres dzielnika
B - długość dzielnika
C - długość dzielnej
Parametry wyjściowe: HL - adres ilorazu

Uwagi: Realizowane jest mnożenie dwóch liczb o różnej długości, nie przekraczającej 128 bajtów. Powstająca reszta jest umieszczana w polu przed ilorazem i ma długość odpowiadającą długości dzielnika. Iloraz umieszczany jest w polu dzielnej. Jeśli dzielnik jest zerowy, sygnalizowany jest nadmiar.

6. MSIG - podprogram sumowania N liczb zmiennoprzecinkowych (FUN=5).

Parametry wejściowe: HL - adres pierwszego składnika
B - liczba składników
C - długość składników - 1
Parametry wyjściowe: DE - adres wyniku

Uwagi: Sumowanych jest N liczb zmiennoprzecinkowych o tej samej długości umieszczonych kolejno od adresu wskazywanego przez HL . Suma o długości pojedynczego składnika umieszczana jest w polu wskazywanym przez DE.

7. RUN - podprogram wykonujący program współbieżnie (FUN=6).

Parametry wejściowe: HL - adres pierwszego argumentu
DE - adres drugiego argumentu
B - długość argumentów - 1
C - funkcja wywoływana przez program
A - liczba argumentów
Parametry wyjściowe: HL - adres wyniku

Uwagi: Program dzieli zadania między aktywne procesory, następnie równolegle oblicza zadaną funkcję.

8. MADF - podprogram dodawania zmiennoprzecinkowego (FUN=7).

Parametry wejściowe: HL - adres pierwszego składnika
DE - adres drugiego składnika
C - długość argumentów - 1
Parametry wyjściowe: HL - adres sumy

9. MSUF - podprogram odejmowania zmiennoprzecinkowego (FUN=8).

Parametry wejściowe: HL - adres odjemnej
DE - adres odjemnika
C - długość argumentów - 1
Parametry wyjściowe: HL - adres różnicy

10. MMUF - podprogram mnożenia zmiennoprzecinkowego (FUN=9).

Parametry wejściowe: HL - adres mnożnej
DE - adres mnożnika
C - długość argumentów - 1
Parametry wyjściowe: HL - adres iloczynu

11. MDIF - podprogram dzielenia zmiennoprzecinkowego (FUN=10).

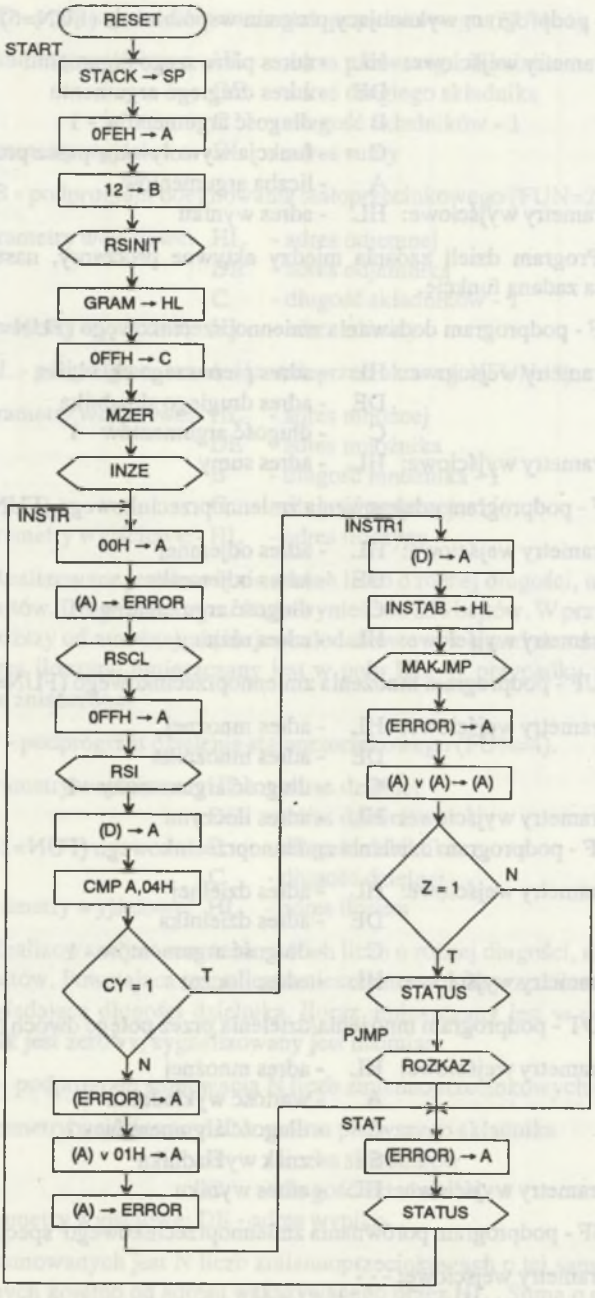
Parametry wejściowe: HL - adres dzielnej
DE - adres dzielnika
C - długość argumentów - 1
Parametry wyjściowe: HL - adres ilorazu

12. MMDT - podprogram mnożenia/dzielenia przez potęgę dwóch (FUN=11).

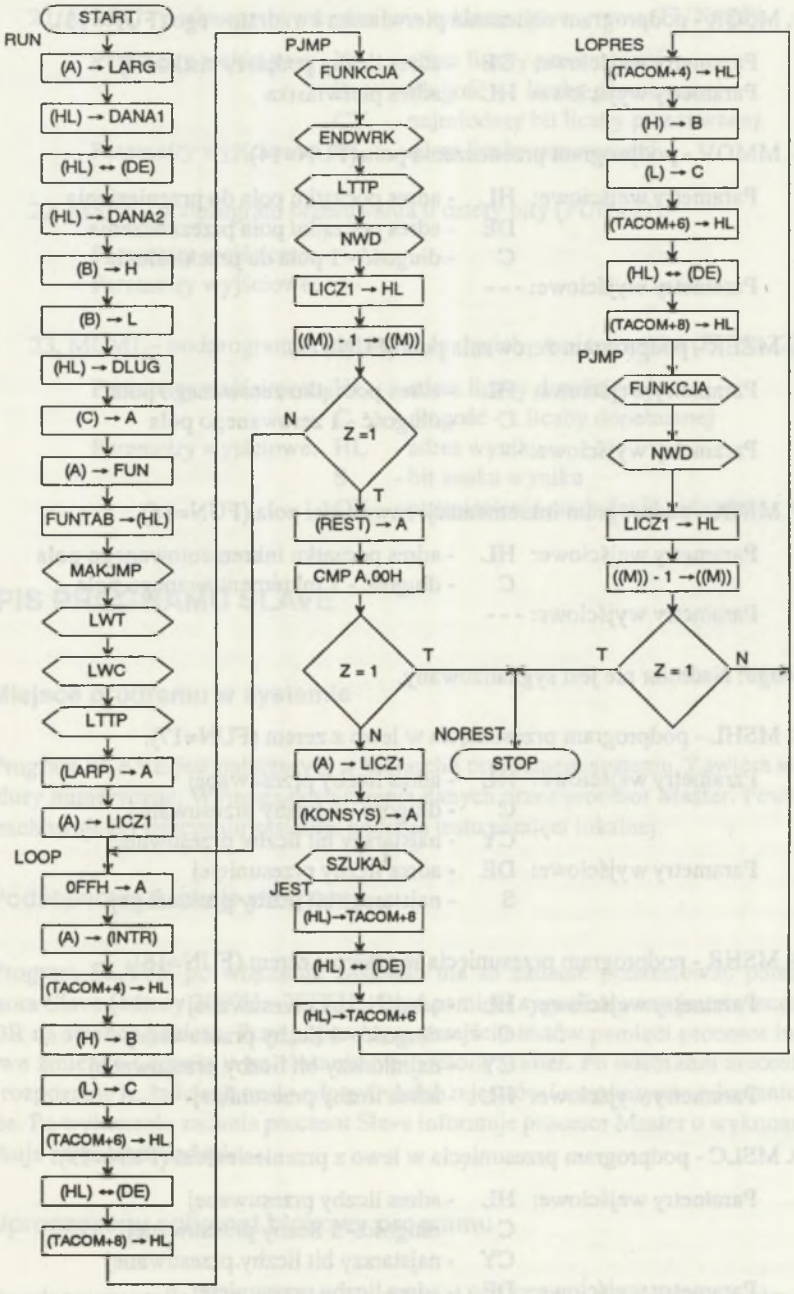
Parametry wejściowe: HL - adres mnożnej
A - wartość wykładnika
C - długość argumentów - 1
S - znak wykładnika
Parametry wyjściowe: HL - adres wyniku

13. MCSF - podprogram porównania zmiennoprzecinkowego specjalnego (FUN=12).

Parametry wejściowe: - - -
Parametry wyjściowe: - - -



Rys. 7. Schemat blokowy głównej pętli programu Master
Fig. 7. Block diagram of the main loop of the Master programme



Rys. 8. Schemat blokowy procedury pracy równoległej
Fig. 8. Block diagram of the procedure of parallel operation

14. MSQR - podprogram obliczania pierwiastka kwadratowego (FUN=13).

Parametry wejściowe: DE - adres liczby podpierwiastkowej
Parametry wyjściowe: HL - adres pierwiastka

15. MMOV - podprogram przenoszenia pola (FUN=14).

Parametry wejściowe: HL - adres początku pola do przeniesienia
DE - adres początku pola przeznaczenia
C - długość - 1 pola do przeniesienia
Parametry wyjściowe: - - -

16. MZER - podprogram zerowania pola (FUN=15).

Parametry wejściowe: HL - adres początku zerowanego pola
C - długość - 1 zerowanego pola
Parametry wyjściowe: - - -

17. MINX - podprogram inkrementacji zawartości pola (FUN=16).

Parametry wejściowe: HL - adres początku inkrementowanego pola
C - długość - 1 inkrementowanego pola
Parametry wyjściowe: - - -

Uwaga: Nadmiar nie jest sygnalizowany.

18. MSHL - podprogram przesunięcia w lewo z zerem (FUN=17).

Parametry wejściowe: HL - adres liczby przesuwanej
C - długość - 1 liczby przesuwanej
CY - najstarszy bit liczby przesuwanej
Parametry wyjściowe: DE - adres liczby przesuniętej
S - najstarszy bit liczby przesuniętej

19. MSHR - podprogram przesunięcia w prawo z zerem (FUN=18).

Parametry wejściowe: HL - adres liczby przesuwanej
C - długość - 1 liczby przesuwanej
CY - najmłodszy bit liczby przesuwanej
Parametry wyjściowe: HL - adres liczby przesuniętej

20. MSLC - podprogram przesunięcia w lewo z przeniesieniem (FUN=19).

Parametry wejściowe: HL - adres liczby przesuwanej
C - długość - 1 liczby przesuwanej
CY - najstarszy bit liczby przesuwanej
Parametry wyjściowe: DE - adres liczby przesuniętej
S - najstarszy bit liczby przesuniętej

21. MRRC - podprogram przesunięcia cyklicznego w prawo (FUN=20).

Parametry wejściowe: HL - adres liczby przesuwanej
B - długość - 1 liczby przesuwanej
CY - najmłodszy bit liczby przesuwanej
Parametry wyjściowe: HL - adres liczby przesuniętej

22. MSRF - podprogram przesuwania o cztery bity (FUN=21).

Parametry wejściowe: - - -
Parametry wyjściowe: - - -

23. MCML - podprogram dopełnienia do dwóch - zmiana znaku (FUN=22).

Parametry wejściowe: HL - adres liczby dopełnianej
C - długość - 1 liczby dopełnianej
Parametry wyjściowe: HL - adres wyniku
S - bit znaku wyniku
CY - przeniesienie po dodaniu jedności

6. OPIS PROGRAMU SLAVE

6.1. Miejsce programu w systemie

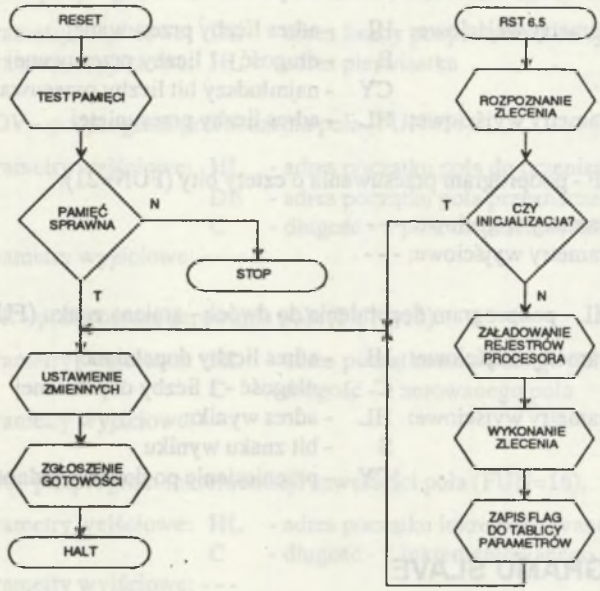
Program SLAVE jest najniższym w hierarchii programem systemu. Zawiera standardowe procedury numeryczne. Wymaga dostarczenia danych przez procesor Master. Pewną "autonomię" zachowuje po włączeniu zasilania podczas testu pamięci lokalnej.

6.2. Podstawowe funkcje programu

Program SLAVE po włączeniu zasilania ma za zadanie przetestować pamięć lokalną procesora Slave (adresy 2000H - 7FFFH). Błąd pamięci sygnalizowany jest świeceniem diody ERROR na ściance pakietu. Przy poprawnym przejściu testów pamięci procesor inicjuje podstawowe zmienne i oczekuje na zlecenia z procesora Master. Po odebraniu zlecenia procesor Slave rozpoznaje je, ładuje dane do odpowiednich rejestrów i rozpoczyna wykonanie zleconego zadania. Po wykonaniu zadania procesor Slave informuje procesor Master o wykonaniu zadania i oczekuje na kolejne zadanie.

6.3. Uproszczony schemat blokowy programu

Po włączeniu zasilania generowany jest sygnał RESET, procesor przeprowadza test pamięci. Gdy test wypadnie negatywnie, rozpoczyna się wykonywanie jałowej pętli. Dioda ERROR



Rys. 9. Uproszczony schemat blokowy programu Slave
Fig. 9. Simplified block diagram of the SLAVE programme

pozostaje zapalona. W przypadku pozytywnego zakończenia testu ustawiane są zmienne programu, zgłaszana jest gotowość procesora Slave do wykonania zadania. Następnie procesor oczekuje na następne zadanie, przechodząc w stan HALT. Procesor Master informuje procesor Slave o nowym zadaniu generując przerwanie 6.5. Dla mikroprocesora 8085 oznacza to zaczęcie wykonywania programu od adresu 0034H. W pierwszym rzędzie następuje rozpoznanie zlecenia. W przypadku wykrycia zlecenia inicjalizacji przeprowadzane jest działanie jak po poprawnym przejściu testu pamięci . Po rozpoznaniu innego polecenia następuje ładowanie rejestrów procesora zawartością tablicy parametrów i wykonanie polecenia. Po zakończeniu zadania flagi procesora zapamiętywane są w tablicy parametrów. Następnie ma miejsce ustawienie zmiennych oraz zgłoszenie gotowości procesora do wykonania następnego zadania. Procesor przechodzi w stan oczekiwania na następne zadanie.

6.4. Obsługa programu SLAVE

Program SLAVE korzysta z danych umieszczonych w tablicy parametrów, której adres przekazywany jest przez procesor Master jako pierwsze zlecenie dla procesora Slave. Opis tablicy parametrów (TACOM) zamieszczono w rozdziale 5.4.2.

6.5. Opis techniczny programu Slave

Program Slave realizuje następujące funkcje:

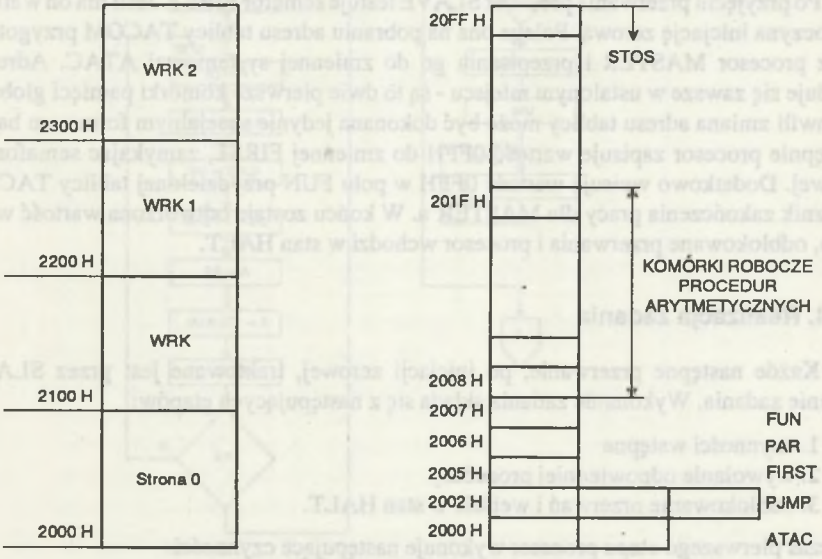
1. test pamięci lokalnej RAM;
2. inicjalizację wewnętrzną;
2. inicjalizację zewnętrzną;
4. pobranie tablicy TACOM;
5. działania arytmetyczne na liczbach wielokrotnej precyzji.

6.5.1. Organizacja pamięci

Każdy procesor Slave posiada 8 kB pamięci EPROM i 24 kB pamięci RAM. Obszar pamięci RAM podzielony został następująco:

- strona zerowa (256B);
- trzy obszary robocze procedur arytmetycznych o długości równej najdłuższym spodziewanym argumentom (3 x 256B).

Strona zerowa zawiera zmienne systemowe i komórki robocze procedur arytmetycznych. Pozostały obszar pamięci RAM może być wykorzystany do ładowania dodatkowych procedur w postaci programów (patrz funkcja numer 5 programu Slave).



Rys. 10. Organizacja pamięci RAM procesora SLAVE
Fig. 10. Organisation of the RAM memory of the SLAVE processor

6.5.2. Inicjalizacja wewnętrzna

Po sygnale RESET rozpoczyna się realizacja testu pamięci lokalnej RAM. Początkowo zostaje chwilowo zaświecony wskaźnik ERROR sygnalizując jego sprawność i poprawne zerowanie procesora. Następnie dioda ERROR gaśnie i rozpoczyna się realizacja testu. Test zapisuje kolejne komórki pamięci lokalnej wzorcem inkrementowanym, po czym następuje odczyt całej zapisanej pamięci. W przypadku stwierdzenia niezgodności wzorca wpisanego i odczytanego zapalana jest ponownie dioda ERROR i wstrzymana realizacja programu. Po pomyślnym wykonaniu testu program wykonuje następujące czynności:

1. zeruje komórkę FIRST stanowiącą semafor inicjacji zewnętrznej;
2. wstawia do komórki PJMP kod rozkazu skoku (0C3H);
4. ustawia wskaźnik stosu na koniec strony zerowej;
5. rozkazem SIM maskuje wszystkie przerwania poza RST 6.5;
6. odblokowuje przerwania;
7. wchodzi w stan HALT.

Mikroprocesor może wyjść ze stanu HALT tylko przez przyjęcie przerwania lub RESET. Stan oczekiwania na przerwanie objawia się zgaszeniem wszystkich wskaźników na płycie czołowej pakietu.

6.5.3. Inicjalizacja zerowa - zewnętrzna

Po przyjęciu przerwania procesor SLAVE testuje semafor FIRST. Jeśli ma on wartość 00H, rozpoczyna inicjację zerową. Polega ona na pobraniu adresu tablicy TACOM przygotowanego przez procesor MASTER i przepisaniu go do zmiennej systemowej ATAC. Adres tablicy znajduje się zawsze w ustalonym miejscu - są to dwie pierwsze komórki pamięci globalnej. Od tej chwili zmiana adresu tablicy może być dokonana jedynie specjalnym formatem bajtu FUN. Następnie procesor zapisuje wartość 0FFH do zmiennej FIRST, zamykając semafor inicjacji zerowej. Dodatkowo wpisuje wartość 0FFH w polu FUN przydzielonej tablicy TACOM. Jest znacznik zakończenia pracy dla MASTER'a. W końcu zostaje odtworzona wartość wskaźnika stosu, odblokowane przerwania i procesor wchodzi w stan HALT.

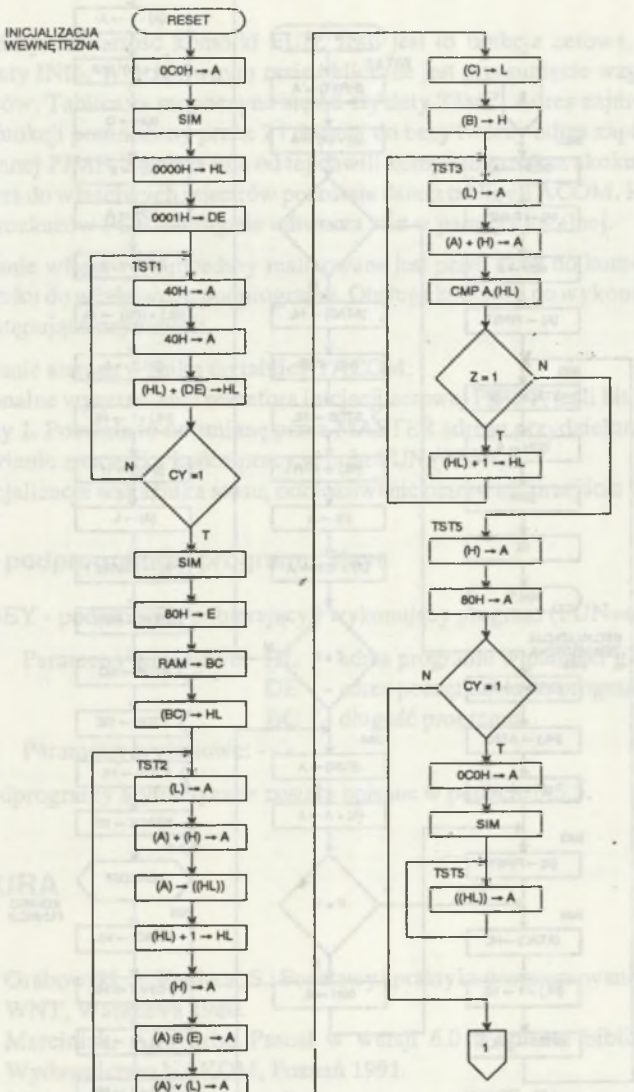
6.5.4. Realizacja zadania

Każde następne przerwanie, po inicjacji zerowej, traktowane jest przez SLAVE jako zlecenie zadania. Wykonanie zadania składa się z następujących etapów:

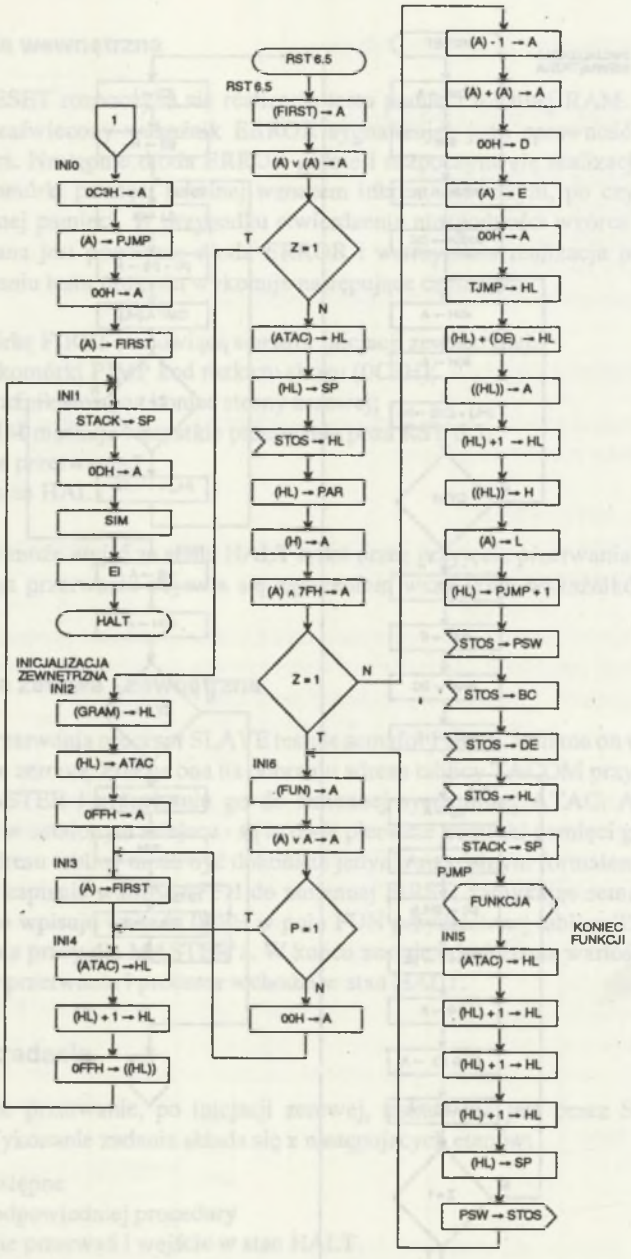
1. czynności wstępne
2. wywołanie odpowiedniej procedury
3. odblokowanie przerwań i wejście w stan HALT.

Podczas pierwszego etapu procesor wykonuje następujące czynności:

1. Ustawia wskaźnik stosu na początek tablicy TACOM. Rozkazem POP pobiera dane z komórek PAR i FUN i przepisuje do zmiennych systemowych.



Rys.11.Schemat blokowy programu Slave cz. 1
Fig. 11. Block diagram of the SLAVE programme p. 1



Rys.11.Schemat blokowy programu Slave cz. 2
Fig. 11. Block diagram of the SLAVE programme p. 2

2. Analizuje zawartość komórki FUN. Jeśli jest to funkcja zerowa, następuje skok do etykiety INI6. W przeciwnym razie obliczane jest przesunięcie względem bazy tablicy adresów. Tablica ta rozpoczyna się od etykiety TJMP. Adres zajmuje 2 bajty, stąd też kod funkcji pomnożony przez 2 i dodany do bazy tworzy adres zapisywany do obszaru zmiennej PJMP. Zawiera ona od tej chwili kompletny rozkaz skoku.
3. Pobiera do właściwych rejestrów pozostałe dane z tablicy TACOM. Realizuje tę operację ciąg rozkazów POP. Następnie odtwarza stos w pamięci lokalnej.

Wywołanie właściwej procedury realizowane jest przez skok do komórki PJMP zawierającej adres skoku do właściwego podprogramu. Obsługa końcowa po wykonaniu danego zadania obejmuje następujące czynności:

1. zapisanie statusu wyniku do tablicy TACOM;
2. opcjonalne wyzerowanie semafora inicjacji zerowej FIRST, jeśli bit 7 komórki FUM był równy 1. Pozwala to na zmianę przez MASTER adresu przydzielonej tablicy TACOM;
3. ustawienie znacznika końca pracy w polu FUN (0FFH);
4. reinicjalizacja wskaźnika stosu, odblokowanie przerwań, przejście w stan HALT.

6.5.5. Opis podprogramów programu Slave

OBEY - podprogram pobierający i wykonujący program (FUN=6).

Parametry wejściowe: HL - adres programu w pamięci globalnej
DE - adres początku/startu programu w pamięci lokalnej
BC - długość programu
Parametry wyjściowe: - - -

Pozostałe podprogramy arytmetyczne zostały opisane w punkcie 5.5.5.

LITERATURA

[1] Grabowski, J., Koślacz, S.: Podstawy i praktyka programowania mikroprocesorów, WNT, Warszawa 1980.
[2] Marciniak, A.: Turbo Pascal w wersji 6.0 z opisem biblioteki Turbo Vison, Wydawnictwo NAKOM, Poznań 1991.
[3] Misiurewicz, P.: Układy mikroprocesorowe, WNT, Warszawa 1983.
[4] Świniarski, R.: System Operacyjny CP/M, WNT, Warszawa 1988.
[5] Turbo Assembler Version 1.0 Reference Guide, Borland, Scotts Valley 1990.
[6] Turbo Assembler Version 1.0 User's Guide, Borland, Scotts Valley 1990.
[7] Turbo Pascal Version 6.0 Library Reference, Borland, Scotts Valley 1990.
[8] Turbo Pascal Version 6.0 Programmers's Guide, Borland, Scotts Valley 1990.
[9] Turbo Pascal Version 6.0 User's Guide, Borland, Scotts Valley 1990.

Wpłynęło do Redakcji w czerwcu 1994 r.

Abstract

The article presents the software of a multiprocessor system with a modular construction based on the Intel 8085 microprocessor. The software is divided hierarchically into three levels: the software of the supervising system, the software of the distinguished processor - MASTER and others processors - SLAVE. The structure of consecutive software blocks is discussed. The technical description of MASTER and SLAVE programs are presented as well as the description of the used mathematical and numerical procedures. The article presents also the methodology of developing programs and developing tasks for the constructed system. The described solutions have been designed and implemented at the Division of Digital Equipment and Microprocessors of the Institute of Electronics of the Silesian Technical Univer.ity in Gliwice.