1 Podstawy Techniki Mikroprocesorowej

1.1 Klasyfikacja komputerów

- **SISD** – Single Instruction Single Data
- **SIMD** – Single Instruction Multiple Data
- **MISD** – Multiple Instruction Single Data
- **MIMD** – Multiple Instruction Multiple Data

1.2 Rola elementów składowych

- **Procesor** – procesor to jednostki arytmetyczno-logiczne, sterują reprezentacją danych. Posiadają rejestry ogólnego przeznaczenia oraz rejestry specjalne jak np. licznik rozkazów, wskaźnik stosu czy rejestr flag. Ma zintegrowany system przerwań, oferuje pracę wielozadaniową. Możliwa jest do wyboru jedna z dwóch organizacji listy rozkazów - RISC i CISC
- **Pamięć** – wyróżniamy pamięć programu i pamięć danych. Do rozdziału zasobów możemy wykorzystać albo stronicowanie, albo segmentację. Część pamięci może służyć jako cache – do przechowywania wyników już wcześniej wykonanych operacji.
- **Układy wejściowo-wyjściowe** – standardowe porty, często korzystające z technologii DMA. Służą do zbierania bądź oddelegowywania sygnałów z jednostki centralnej.

1.3 Najważniejsze elementy procesora

- Jednostka centralna
- Zegar
- Wewnętrzna pamięć programu ROM
- Wewnętrzna pamięć danych RAM
- Układy czasowo-licznikowe
- System przerwań
- Szeregowe wejścia-wyjścia
- Wejścia-wyjścia

1.4 Architektura 8051

- Jednostka centralna w organizacji **CISC** (111 rozkazów, w tym rozkazy 1, 2 i 3-bajtowe). Do operacji logicznych i prostych arytmetycznych wystarczą jeden lub dwa cykle maszynowe. Do mnożenia i dzielenia potrzebne są cztery cykle.
- Wewnętrzny rezonator kwarcowy o częstotliwości 12 MHz.
- Wewnętrzna pamięć programu ROM 8-bitowa 4KW.
- Wewnętrzna pamięć danych RAM 128B (czasem 256B), zewnętrzna do 64KW.
- Dwa 16-bitowe liczniki sterowane wewnętrznie lub zewnętrznie, 4 tryby pracy.
- 2-poziomowy system przerwań z obsługą integralnych elementów dodatkowych.
- **Brak DMA**.

1.5 Przetwarzanie rozkazu – fazy

1. Pobranie rozkazu
2. Dekodowanie rozkazu
3. Wyznaczenie adresu
4. Załadowanie argumentów
5. Wykonanie rozkazu
6. Zapisanie rezultatów w pamięci

Kroki **1**, **3**, **4** i **6** korzystają z pamięci.

1.6 Podstawowe tryby adresowania

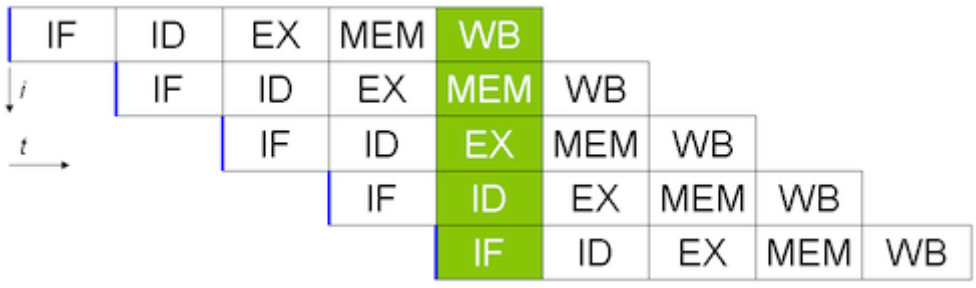
- Rejestrowy (bezpośredni) – argumenty i wyniki operacji znajdują się z rejestrach procesora, bardzo szybki tryb – 3. to wskazanie rejestru i 4. szczątkowe załadowanie argumentów.
- Rejestrowy (pośredni) – zawartość rejestru adresuje pamięć (układy I/O), tryb ten jest wolniejszy niż bezpośredni rejestrowy.
- Bezpośredni – adres po rozkazie pokazuje gdzie leżą argumenty, 3. jest proste w realizacji, natomiast załadowanie argumentów trwa długo.
- Natychmiastowy – argumenty „lecą" razem z rozkazem, nie ma potrzeby wyznaczania adresu i załadowania argumentów, czasem znika też potrzeba zapisania rezultatów.
- Bazowo-indeksowy – bardzo złożony tryb adresowania.

1.7 Przetwarzanie potokowe

Przetwarzanie potokowe jest jednym ze sposobów sekwencyjnego (szeregowego) przetwarzania danych. Sposób działania:

- cykl przetwarzania dzieli się na odrębne bloki przetwarzania, z których każdy oprócz ostatniego jest połączony z następnym
- dane po przejściu przez jeden blok trafiają do następnego, aż osiągną ostatni blok

Dzięki temu, że przetwarzanie odbywa się w rozdzielonych blokach, każdy z nich może wnieść swoją logikę (np. 1. blok sortuje dane, 2. usuwa sąsiadujące duplikaty) bez konieczności wbudowywania jej na poziomie całego systemu. Bloki-następniki są zależne od pracy swoich (niekoniecznie bezpośrednich) poprzedników.



Rysunek I: Uproszczony schemat potokowości. Rozkazy zaznaczone na zielono wykonywane są równocześnie.

1.8 Procedury i funkcje – rola stosu

Stos to obszar w pamięci wydzielony dla danego wątku, służący do przechowywania adresów powrotu i zmiennych lokalnych. Okna rejestrowe są używane do wspierania wywołań procedur w taki sposób, aby były one widziane jako cache w zawartości stosu.

1.9 System przerwań – sekwencja działania

W momencie, gdy procesor przyjmie zgłoszenie przerwania, wykonywane są następujące działania:

1. Identyfikacja linii urządzenia wysyłającego przerwanie – ze względu na różne priorytety przerwań (arbitraż programowy, arbitraż sprzętowy, tj. równoległy, szeregowy bądź mieszany).
2. Zapamiętanie stanu systemu – rejestry specjalne i ogólne zapamiętywane są albo na stosie (**przełączenie danych**) albo w innym banku pamięci (**przełączenie kontekstu**) w sposób automatyczny.
3. Wykonanie procedury obsługi z adresu uzyskanego podczas identyfikacji źródła.

1.10 System przerwań 8051

W układzie 8051 jest zaimplementowany system wieloźródłowy przerwań o dwóch priorytetach z możliwością zagnieżdżania. Mikrokomputer 8051 akceptuje żądania przerwań z sześciu źródeł:

- Urządzenie zewnętrzne na linii **INT0** – znacznik **IE0**
- Urządzenie zewnętrzne na linii **INT1**
- Trzy układy czasowo-licznikowe – przepełnienie licznika **T0**, **T1** lub **T2** (znaczniki **TF0**, **TF1** i **TF2**)
- Jedno przerwanie od portu szeregowego – koniec nadawania znaku **TI** lub koniec odbierania **RI**

Podczas cyklu przyjęcia przerwania są wykonywane następujące operacje:

1. Ustawienie wewnętrznego przerzutnika poziomu przerwania
2. Zapisanie na stosie zawartości licznika rozkazów PC
3. Wpisanie do licznika rozkazów adresu początku programu obsługi przerwania:
   - `0003H` – dla przerwania zewnętrznego INT0
   - `000BH` – dla przerwania z licznika T0
   - `0013H` – dla przerwania zewnętrznego INT1
   - `001BH` – dla przerwania z licznika T1
   - `0023H` – dla przerwania z portu szeregowego
   - `002BH` – dla przerwania z licznika T2

1.11 DMA – zasady i tryby pracy

DMA (*Direct Memory Access*) służy do tego, aby odciążyć procesor z operacji na pamięci w przypadku, gdy chodzi tylko o przesyłanie z jednego urządzenia do drugiego. Przykładowo, gdy chcemy z pamięci wysłać coś przez modem, procesor musiałby pobrać te dane z pamięci i wysłać do modemu. DMA robi to za niego, dzięki czemu procesor może zająć się w tym momencie czymś innym.

DMA nie jest w stanie samodzielnie „myśleć", czyli jeśli trzeba wykonać na danych dodatkowe operacje, to musi już zająć się tym procesor. DMA tylko sobie spokojnie czeka na dane, a gdy dostanie zlecenie (musi mieć określone źródło i cel transmisji, samodzielnie DMA nie jest w stanie nic wymyślić), to bierze się do roboty.

Warto zauważyć, że DMA konkuruje z procesorem o magistralę (co jest zrozumiałe, bo musi jakąś drogą te dane przesłać i pobrać), jednak upomina się o nią tylko, gdy dostanie zadanie do wykonania.

Tryby pracy:

- **Blokowy** – DMA dostaje zlecenie na przesłanie danych, to je przesyła. Wszystkie na raz. W związku z tym procesor nie ma dostępu do magistrali, więc musi siedzieć cicho i czekać, aż wszystko się prześle. Oczywiście, może sobie coś niecoś obliczać samodzielnie w tym czasie, ale bez dostępu do urządzeń zewnętrznych zwykle niewiele może zrobić. Za to transmisja w tym czasie jest najszybsza.
- **Wykradanie taktów** – procesor ustala, kiedy DMA może wysyłać dane, a kiedy nie. Dzięki temu może on korzystać z magistrali wtedy, kiedy tylko potrzebuje, ale DMA musi dłużej czekać.
- **Zgodnie z zapotrzebowaniem** – ustalane jest na początku to, ile danych DMA może na raz przesłać, i kiedy magistrala jest wolna, to DMA sobie przesyła. Po przesłaniu danej porcji (czyli bloku), jeśli procesor nie zażąda dostępu do magistrali, to przesyłany jest następny blok.

1.12 Pamięć

1.12.1 Klasyfikacja pamięci ROM

- **ROM**
  - Różne typy organizacji
  - Najszybsze spośród wszystkich
  - Mogą być o różnych pojemnościach
  - Zawsze gotowa do odczytu
  - Nie może zostać wyczyszczona - dane w niej są zapisywane na etapie produkcji i nic z nimi później nie można zrobić.
  - Produkowane w technologii MOS
  - Zalety: niska cena przy dużej liczbie egzemplarzy, duże pojemności, zachowuje informacje po odłączeniu zasilania, mały pobór mocy
  - Wady: brak możliwości modyfikacji zawartości, wysoki koszt opracowania pamięci o nowej zawartości informacji
- **PROM**
  - Takie, jak ROM, ale:
  - Różne pojemności, ale głównie małe
  - Można je programować, ale tylko raz
  - Programowanie off-line - trzeba użyć specjalnego programatora, który „wypala" dane w układzie
  - Technologia TTL
  - Zalety: łatwość wprowadzania informacji przez użytkownika, duża szybkość
  - Wady: bardzo ograniczone możliwości modyfikacji, duży pobór mocy
- **EPROM**
  - Wolne
  - Pojemności mogą być duże
  - Zawsze gotowa do odczytu
  - Może być wiele razy programowana
  - Zbudowana w technologii tranzystorów z pływającą bramką
  - Programowanie off-line w obecności ultrafioletu - działa to tak, że układ może być programowany tylko, jeśli świeci na niego światło ultrafioletowe przez okienko na górze, ze względu na wykorzystanie zjawiska fotoelektrycznego zewnętrznego
  - Używany tam, gdzie dane się zmieniają
  - Zalety: łatwość wielokrotnego wprowadzania informacji, mały pobór mocy, umiarkowana cena
  - Wady: możliwość przypadkowego kasowania światłem rozproszonym, ograniczona liczba kasowań
- **EEPROM**
  - Wolne
  - Duże pojemności
  - Zawsze gotowa do odczytu
  - Może być wiele razy programowana
  - Programowanie on-line - tzn nie wymaga specjalnych programatorów
  - Kasowanie i programowanie dokonuje się elektrycznie
  - Kasowanie danych blokowe
  - Zalety: możliwość wprowadzania i modyfikacji informacji w układzie aplikacyjnym, mały pobór mocy
  - Wady: wysoka cena, długi czas zapisu informacji

- **EAPROM**
  - Wolne
  - Duże pojemności
  - Zawsze gotowa do użytku
  - Może być wiele razy programowana
  - Kasowanie i programowanie elektryczne
  - Możliwe jest kasowanie wybiórcze

1.12.2 Klasyfikacja pamięci RAM

- **SRAM**
  - Szybkie
  - Różne pojemności
  - Wykorzystuje przerzutniki bistabilne - czyli takie, gdzie oba stany są
  - stabilne, ale do zmiany stanu wymagany jest zewnętrzny sygnał wyzwalający.
  - Wymaga prądu do przechowywania informacji, po utracie zasilania tracimy informację
  - Nie wymaga odświeżania
  - Pamięć półprzewodnikowa
  - Zalety: bardzo łatwe i szybkie zapisywanie informacji, duże szybkości i pojemności
  - Wady: dla zachowania informacji wymaga ciągłego zasilania
- **DRAM**
  - Szybkie
  - Różne pojemności
  - Tak, jak SRAM, wymaga prądu do utrzymywania danych
  - Wykorzystuje kondensatory do przechowywania informacji
  - Konieczne jest odświeżanie!
  - Pamięć półprzewodnikowa
  - Zalety: bardzo łatwe i szybkie zapisywanie informacji, niskie ceny, duże pojemności
  - Wady: dla zachowania informacji wymaga ciągłego zasilania i okresowego odświeżania

1.12.3 Cykle odczytu i zapisu

1. **Odczyt ROM/SRAM**
   Pierwszą fazą jest ustawienie adresu zczytanego w strukturze pamięci (linie adresowe). Uaktywnienie linii **MEMREQ** na podstawie adresu, a także linii CE (logicznego wyboru danego układu pamiętającego) – stanem niskim. Następnie uaktywnia się linię odczytu i pobiera dane
2. **Zapis SRAM**
   W zapisie do pamięci RAM adres i wybór układu pozostaje jak w odczycie. Zamiast linii odczytu (RD) ustawiamy stan niski na linii zapisu (WR) i pobieramy dane od procesora, który musi udostępnić je na tak długi czas, aby zapis został ukończony.
3. **Odczyt DRAM**
   Pierwszą fazą żądania odczytu z pamięci stanowi pobranie adresu wiersza, w którym znajduje się komórka i zatwierdzeniu tego adresu, po ustabilizowaniu się stanu szyny adresowej sygnałem RAS (*Row Address Strobe*). Następnie na szynę adresową pamięci podawany jest adres kolumny zawierającej żądaną komórkę, który zatwierdzany jest sygnałem CAS (*Column Address Strobe*). Odstęp czasu pomiędzy sygnałami CAS i RAS wynika z konstrukcji pamięci i musi zapewnić czas nie tylko na ustabilizowanie się stanu szyny adresowej, lecz także na wysterowanie wiersza do odczytu.
4. **Zapis DRAM**
   Pierwszą fazę zapisu stanowi pobranie adresu, pod który będziemy zapisywać. Następnie następuje synchronizacja RAS wiersza w pamięci i CAS kolumny. Mając pełny adres, wyzwalamy sygnał na linii WR, by ostatecznie zapisać dane.

1.12.4 Odświeżanie

Odświeżanie DRAM jest potrzebne ze względu na sposób przechowywania danych – są one trzymane w kondensatorach, a jak wiadomo, ich ładunek z czasem się traci. Nie byłoby potrzeby odświeżania pamięci, gdybyśmy zawsze używali całej pamięci. Jednakże, nasz zapis czy odczyt musi nastąpić w to samo miejsce. Odświeżanie może niebezpiecznie „dociążyć" procesor.

Wyróżniamy trzy sposoby odświeżania:

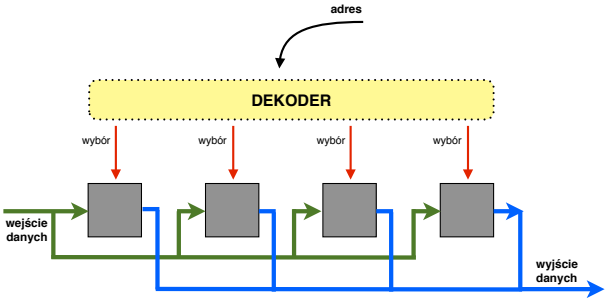
- tylko sygnałem RAS (*RAS only refresh*) – tylko adres wiersza zostaje zatwierdzony po sygnale RAS. Nie ma potrzeby używania CAS, ale potrzebny jest zewnętrzny licznik do iteracji po wierszu.
- ukryte – użycie obu linii CAS i RAS, najpierw następuje synchronizacja wiersza, potem kolumny.
- CAS przed RAS – gdy brak sygnału na /CAS przed /RAS, DRAM ignoruje adresy i używa wewnętrznego licznika.

1.12.5 Specjalne tryby pracy

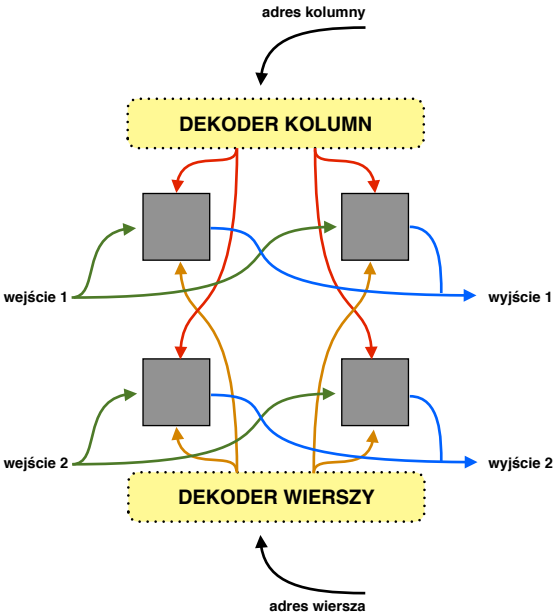
- **odczyt-modyfikacja-zapis** – adres kolumny nie musi zostać zestrobowany, można natomiast użyć adresu z impulsów /CAS niskiego potencjału i następnie dokonać zapisu w przeciągu kilku nanosekund.
- **tryb stronicowy** – wiersz DRAM pozostaje otwarty przez utrzymywanie niskiego potencjału /RAS w trakcie trwania odczytów bądź zapisów odosobnionym impulsem CAS.
- **tryb nakładkowy** – cztery impulsy CAS mają dostęp do czterech sekwencyjnych lokacji w wierszu.

1.12.6 Budowa bloków

Aby zorganizować komórki pamięci w sprawnie funkcjonujący układ, należy je odpowiednio zaadresować. Najprostszym sposobem jest zorganizowanie pamięci liniowo, tj. **adresowanie 2D**. Do każdej komórki podłączone jest wejście, sygnał wybierania pochodzący z dekodera i wyjście. Nieco innym sposobem jest adresowanie przy pomocy **matrycy 3D**, gdzie pamięć organizuje się dzieląc dostępne elementy na wiersze i kolumny. Dostęp do pojedynczego elementu pamiętającego można uzyskać po zaadresowaniu odpowiedniego wiersza i kolumny, dlatego też komórka RAM obok wejścia i wyjścia musi dysponować jeszcze dwoma sygnałami wybierania, odpowiednio z dekodera kolumn i wierszy.



Rysunek II: Przykład organizacji 2D



Rysunek III: Przykład organizacji 3D

1.13 Pamięć w 8051

Mikrokontroler 8051 ma dwie odrębne przestrzenie adresowe pamięci: 64kB pamięci programu i 64kB pamięci danych. Pamięć programu może być tylko odczytywana, nie może być zapisywana podczas normalnej pracy systemu, pamięć danych może być odczytywana i zapisywana. Mikrokontroler pobiera rozkazy z wewnętrznej pamięci programu gdy na wejściu EA jest jedynka logiczna, a po przekroczeniu adresu `0FFFh` (4kB), z zewnętrznej pamięci programu. Gdy na wejściu jest zero, mikrokontroler pobiera wyłącznie z zewnętrznej pamięci programu (wymagane użycie sygnału sterującego /PSEN).

Wewnętrzna pamięć danych obejmuje przestrzeń adresową 256 bajtów – adresy 00-7Fh (tzw. obszar „lower" dostępny w trybie adresowania bezpośredniego i pośredniego) to 128 bajtów właściwej wewnętrznej pamięci danych (rejestry R0-R7, stos), adresy 80-FFh to obszar rejestrów specjalnych *Special Function Registers* (ACC, SP, P0-P3).

Większość rejestrów jednostki centralnej mikrokontrolera 8051 jest dostępna w wewnętrznej pamięci danych.

- **Akumulator** – (0E0H) 8-bitowy podstawowy rejestr mikrokontrolera
- **Rejestr B** – 8-bitowa pamięć tymczasowa wykorzystywana np. do operacji dzielenia/mnożenia
- **Rejestry ogólnego przeznaczenia** R0-R7, każdy o rozmiarze 8 bitów. Programista ma dostęp do 4 banków (efektywnie 4×8 rejestrów), określonych bitami RS1:RS0 8-bitowego rejestru znaczników PSW.
- **Rejestr DPTR** – 16-bitowy wskaźnik danych w zewnętrznej pamięci danych
- **Rejestr PC** – 16-bitowy licznik rozkazów
- **Rejestr SP** – 8-bitowy wskaźnik stosu

Mikrokontroler nie ma możliwości zapisu do pamięci programu – w trakcie normalnej pracy systemu zapis możliwy jest tylko do pamięci danych. System uruchomieniowy musi jednak być wyposażony w możliwość zamiany zawartości pamięci programu. Dlatego w konfiguracji magistrali systemowej DSM dla mikrokontrolera 8051 został wprowadzony dodatkowy sygnał PSWR (*Program System Memory Write*), który jest generowany przez kartę monitora, dzięki cezmu można umieścić w pamięci 6264 (MEM8) program dla mikrokontrolera 8051.

W złożonych systemach z 8051 zachodzi potrzeba użycia zewnętrznej pamięci programu i zewnętrznej pamięci danych (mogą być one umieszczone w jednym układzie scalonym, wystarczy zadbać o rozdzielenie przestrzeni adresowej).

1.14 Cache

Cache, czyli pamięć podręczna, służy do tego, aby przechowywać tymczasowo dane dla procesora. Warto zauważyć, że cache jest kompletnie przezroczyste dla procesora - procesor sięga pod dany adres pamięci i nie interesuje się tym, czy dane przyjdą z cache, czy z pamięci.

Istnieje coś takiego jak **hit ratio** i **miss ratio**. Są to współczynniki, odpowiednio, trafień w cache i chybień w cache. Trafienie w cache następuje wtedy, gdy procesor sięga po coś do pamięci, ale owo „coś" znajduje się już w cache, więc w ostateczności zostaje do procesora wysłane z cache, a nie z samej pamięci. Chybienie - wiadomo, danych nie ma w cache, więc muszą zostać pobrane z pamięci.

1.14.1 Budowa i sposoby organizacji

- **Direct Mapping Cache** – dla ustawionego rozmiaru „skoku" `(s)`, tj. odległości relatywnej w pamięci głównej, mamy sytuację taką, że linia `(i)` pamięci Cache może odwoływać się do linii `(i)`, `(s+i)` oraz `(2s+i)` w pamięci głównej.
- **Fully Associative Cache** – każda linia pamięci cache może odwoływać się do dowolnej linii pamięci głównej.
- **Set Associative Cache** – dzielimy pamięć cache na zbiory. Jest to schemat podobny do DMC, jeśli chodzi o skoki. Natomiast w każdym secie możemy dowolnie przydzielić linię z pamięci głównej, z ustalonego zakresu, zdeterminowanego numerem setu.

1.14.2 Mechanizmy zastępowania danych

Zastępowanie danych dzielimy ze względu na fakt uwzględniania informacji z tagów:

- z uwzględnieniem informacji
  - MIN – co najdłużej potrzebne nie będzie. Stosuje się do tego różne dziwne mechanizmy analizujące kod programu (zwłaszcza, jeśli kod programu też leży w cache) sprawdzające, do jakich miejsc w pamięci będziemy się w najbliższym czasie odwoływać. Oczywiście, można łatwo zmylić takie algorytmy poprzez np. rozgałęzienia w programie.
  - LRU – co najdłużej potrzebne nie było. Dość oczywiste, problem polega jednak na tym, że jeśli używamy w kółko np. 10 danych, a w cache jest miejsce na 9, to po pewnym czasie cały czas będziemy mieli chybienia.
- bez uwzględnienia informacji
  - FIFO – pierwsze wlata pierwsze wylata. Problem pojawia się, jeśli np. dużo operujemy na jednej danej z pamięci, a potem obrabiamy inne, korzystając z informacji z tej jednej. Wtedy po chwili zostanie ona wywalona z cache i będzie trzeba ponownie po nią sięgać do pamięci.
  - Random – wbrew pozorom, nie jest to metoda najgorsza! Tego nie było na wykładzie, ale warto zauważyć, że nie wymaga przechowywania jakichkolwiek informacji o tym, kiedy linia została wrzucona do cache bądź kiedy była ostatnio używana, oraz nie wymaga żadnego systemu przewidującego!

Mamy rozwiązane odczytywanie z pamięci za pośrednictwem cache, ale co z zapisywaniem? Oczywiście, sam procesor nie pisze bezpośrednio do pamięci, ale do cache, a następnie cache synchronizuje się z pamięcią główną. Może to robić na trzy sposoby:

- Zawsze, gdy coś jest zapisane do cache, to jest zapisywane również do pamięci głównej. Plus jest taki, że dane na pewno nie zginą, bo są w obu miejscach jednocześnie. Minusem jest powolność.
- Gdy linia jest usuwana z cache, to jest zapisywana w pamięci głównej.
- Gdy linia jest usuwana z cache, to jest zapisywana w pamięci głównej pod warunkiem, że została zmodyfikowana podczas pobytu w cache. To jest przemyślane - ponieważ jeśli nie została zmodyfikowana, to nadpisywalibyśmy w pamięci głównej te same dane na te same dane.

1.15 Stronicowanie i segmentacja

Stronicowanie polega na dopuszczeniu nieciągłości logicznej przestrzeni adresowej procesu. Podstawowa metoda:

- Pamięć fizyczna dzielona na bloki o stałej długości zwane ramkami
- Pamięć logiczna dzielona na bloki o stałej długości zwane stronami
- Rozmiary stron i ramek są identyczne
- Strony z pamięci pomocniczej wprowadzane są w dowolne ramki pamięci operacyjnej

Segmentacja pamięci polega na podzieleniu przez procesor pamięci fizycznej na fragmenty o określonym początku, rozmiarze, atrybutach i identyfikatorze. System tworzy takie segmenty na żądanie aplikacji, przekazując jedynie identyfikatory nie pozwalające na odczytanie parametrów segmentów.

1.16 I/O – urządzenia wejścia/wyjścia

1.16.1 Port równoległy

Intel 8255 – programowalny układ równoległego I/O z 8-bitowymi portami PA i PB. Nadawanie i odbieranie **słowa**. Tryby pracy:

- **tryb 0** – przeznaczony do realizacji bezwarunkowych operacji I/O.
  - dwa porty 8-bitowe i dwa porty 4-bitowe
  - możliwość zaprogramowania każdego portu jako I lub O
  - wyjścia z rejestrami zatrzaskowymi i bez tych rejestrów

- **tryb 1** – przeznaczony do realizacji operacji I/O z przerwaniem, przy jednym kierunku przesyłania danych. Potrzebne do tego celu sygnały są wyprowadzane lub wprowadzane z wykorzystaniem konkretnej linii portu.
- **tryb 2** – przeznaczony do realizacji operacji I/O z przerwaniem, tylko poprzez port A, przy dwóch kierunkach przesyłu danych.

1.16.2 Port szeregowy

Intel 8251 – programowalny układ szeregowego I/O, składający się z:

- rejestru buforowego nadajnika PIPO
- rejestru przesuwnego nadajnika PISO
- rejestru buforowego odbiornika PIPO
- rejestru przesuwnego odbiornika SIPO
- rejestru stanu
- rejestru sterującego
- rejestru służącego do synchronizacji

Informacja przeznaczona do wysłania zapisywana jest do rejestru buforowego nadajnika, stamtąd przepisywana jest do rejestru nadajnika i wysyłana. Informacja odebrana przez układ zapisywana jest do rejestru przesuwnego odbiornika a następnie do rejestru buforowego, skąd może zostać odczytana przez jednostkę centralną.

Tryby pracy:

- **transmisja asynchroniczna** – częstotliwość zegara k razy większa od częstotliwości nadawania, długość znaku to 5-8 bitów, możliwa jest kontrola poprawności transmisji (bit parzystości). Synchronizacja przy użyciu wykrywania przerwy w nadawaniu.
- **transmisja synchroniczna** – częstotliwość k razy większa od nadawania, ilość znaków synchronizacji to 1 lub 2, synchronizacja na dwa sposoby: zewnętrzny i wewnętrzny.

1.16.3 Układy czasowo-licznikowe

Intel 8253 – programowalny timer/licznik, oznaczany często jako PIT, zawierający trzy, niezależne od siebie 16-bitowe liczniki (liczące wstecz) wyposażone we własne wejścia - Gate oraz Clock, jak i wyjścia.

Tryby pracy:

- **Tryb 0** – interrupt on terminal count, odliczanie jednorazowe
- **Tryb 1** – programmable one-shot, generowanie jednego impulsu
- **Tryb 2** – rate generator, dzielnik częstotliwości
- **Tryb 3** – generator z symetrycznym sygnałem wyjścia (jak w trybie 2, ale częstotliwość jest dzielona przez dwa), generator fali prostokątnej
- **Tryb 4** – impuls przerzutnika, sterowany programowo
- **Tryb 5** – impuls przerzutnika, sterowany sprzętowo

1.16.4 Współadresowanie i adresowanie izolowane

- **współadresowanie** – rejestry układów I/O udostępniane są jako zwykłe komórki pamięci w przestrzeni adresowej pamięci.
- **adresowanie izolowane** – rejestry urządzenia dostępne są w odrębnej przestrzeni adresowej zwanej przestrzenią adresową wejścia/wyjścia lub przestrzenią portów; do przesyłania danych do i z takich układów służą rozkazy `OUT` i `IN`.

1.17 Magistrala IEC-625

Interfejs **IEE-488** z 24 wyprowadzeniami. Osiem z nich to wyprowadzenia danych, 3 – linie synchronizacji, 5 – linie sterowania. Robi za kanał komunikacyjny do 16 urządzeń, w logice ujemnej (poziomy TTL). Długość przewodów ok. 20 metrów.

1.17.1 Linie magistrali

- `DIO1-DIO8` – dwukierunkowe linie danych
- `ATN` – kontroler utrzymujący tę linię w stanie niskim podczas transmisji danych
- `DAV` – stan niski sygnalizuje pojawienie się i ważność informacji na szynie danych
- `NDAC` – przyjęcie danych wprowadza linię w stan wysoki
- `NRFD` – stan niski wskazuje, że urządzenie nie jest gotowe do odbioru danych
- `EOI` – wskazuje koniec transmisji
- `REN` – powoduje odłączenie lokalnego sterowania urządzeniem
- `SRQ` – urządzenie, które potrzebuje obsługi wymusza stan niski (żądanie przerwania bieżących zdarzeń)
- `IFC` – zerowanie interfejsu np. po restarcie

1.18 Architektury SIMD

Architektury systoliczne, to architektury specjalizowane do implementacji operacji macierzowych, przetwarzania sygnałów i obrazów w czasie rzeczywistym. Globalnie synchronizowane procesory elementarne są połączone w regularną siatkę. Każdy procesor jest połączony tylko z najbliższymi sąsiadami. Struktura wewnętrzna procesora, w zależności od postawionego zadania, możę być bardzo prosta lub dążyć do stopnia ukomplikowania współczesnych mikroprocesorów. Wyniki przetwarzania są uzyskiwane stopniowo.

1.18.1 Podstawowe struktury

- **tablica z częściowym rozpowszechnianiem danych** – komunikacja PE punkt-punkt oraz magistrala, komunikacja zewnętrzna przez magistralę, rozbudowane układy sterujące, stopień wykorzystania >50%
- **tablica heksagonalna** – komunikacja między procesorami typu punkt-punkt, komunikacja zewnętrzna tylko za pomocą procesorów brzegowych, prosta konstrukcja, duża liczba PE, niski stopień ich wykorzystania (<50%)
- **tablica przetwarzania potokowego** – komunikacja jak w heksagonalnej, prosta konstrukcja, liczba PE mniejsza niż w tablicy heksagonalnej, stopień ich wykorzystania – co najmniej 50%
- **tablica typu wavefront** – asynchroniczna komunikacja punkt-punkt, na zewnątrz tylko brzegowe, dobra skalowalność, łatwe przeprogramowywanie, dobre parametry FTC, stopień wykorzystania PE to 50%
- **tablica z rozpowszechnianiem danych** – komunikacja tylko przez magistrale, rozbudowane układy sterujące, łatwa implementacja algorytmów, wysoki stopień wykorzystania PE

1.18.2 Parametry jakościowe

- **Czas wykonania obliczeń** – czas od rozpoczęcia pierwszej operacji, aż do ukończenia ostatniej
- **Okres przetwarzania potokowego** – odstęp między dwoma sukcesywnymi wynikami obliczeń procesora
- **Okres bloku** – odstęp między inicjacjami dwóch sukcesywnych bloków
- **Wskaźnik użycia procesora**
- **Rozmiar tablicy** – tablica nie może być nieograniczona, czasem musi być ona mniejsza od rzeczywistego rozmiaru problemu, czasem możemy też użyć tablicy jednowymiarowej do zasymulowania tablicy 2D
- **Pamięć lokalna** – używana do rozszerzenia fizycznej tablicy do znacznie większej tablicy wirtualnej

1.18.3 Transputery

Transputer to mikroprocesor w jednym układzie scalonym, zaprojektowany specjalnie do obliczeń równoległych (szybka komunikacja i łatwość połączenia z innymi transputerami). Wraz z nim został opracowany język programowania równoległego OCCAM.

W skład transputera wchodzi procesor typu RISC, wewnętrzna pamięć RAM oraz łącze pamięci zewnętrznej, która umożliwia adresowanie w przestrzeni 4 GB. Do komunikacji z innymi transputerami wykorzystywane są cztery kanały DMA.

1.19 Koprocesor arytmetyczny

Kiedyś koprocesor był osobnym układem dołączonym np. magistralą. Obecnie jest on zintegrowany z procesorem głównym, choć dwoistość architektury nadal jest widoczna – np. każdy z procesorów ma swoje rejestry. Ponadto, nie jest możliwa praca samego koprocesora, nie obsługuje on urządzeń I/O oraz przerwań.

Współpraca polega na tym, że procesor podstawowy pobiera rozkazy zarówno dla siebie, jak i dla koprocesora (mają one specjalną preambułę), dekoduje je i po odcięciu preambuły odsyła rozkaz do koprocesora, przy okazji wyznaczając adresy efektywne argumentów.

1.19.1 Formaty danych

S	E	F	Obiekt
0	MAX	$\neq 0$	plus nieliczba
0	MAX	0	$+\infty$
0	0<E<MAX	$\neq 0$	liczba dodatnia
0	0	$\neq 0$	liczba dodatnia w postaci nieznormalizowanej
0	0	0	+0
1	0	0	–0
1	0	$\neq 0$	liczba ujmena w postaci nieznormalizowanej
1	0<E<MAX	$\neq 0$	liczba ujemna
1	MAX	0	$-\infty$
1	MAX	$\neq 0$	minu nieliczba

Tablica 1: S — bit znaku, E — część wykładnicza, F — część ułamkowa

- format krótki rzeczywisty (*short real*)
- format długi rzeczywisty (*long real*)
- format rzeczywisty rozszerzony (*temporary real*)
- format słowowy całkowity (*word integer*)
- format krótki całkowity (*short integer*)
- format długi całkowity (*long integer*)
- format BCD, upakowany (*packed BCD*)

1.19.2 Rejestry

- **Rejestr znaczników** składa się z ośmiu 2-bitowych pól `TAG0`-`TAG7` zawierających syntetyczne dane o zawartości rejestrów na stosie o znaczeniu:

  00 – liczba zwykła, 01 – zero, 10 – zawartość specjalna (nieskończoność), 11 – rejestr pusty

- Rejestr sterujący

  IC – *infinity control* – określenie modelu nieskończoności, w dwóch trybach: afinicznym (1, wyróżniamy $+\infty$ i $-\infty$) oraz rzutowym (0, tylko jedno ∞).

  RC – *rounding control* – sposób zaokrąglania

  00 – do liczby najbliższej

  01 – w dół

10 – w górę
11 – w kierunku 0

PC – *precision control* – precyzja obliczeń
00 – 24 bity
10 – 53 bity
11 – 64 bity
01 – zarezerwowane

- Rejestr stanu

B – 1 oznacza, że rozkaz niezakończony

TOP – numer rejestru na szczycie stosu

$\mathbf{C}_{3-0}$ – kod warunku

IR – zgłoszenie przerwania: PE (utrata dokładności, UE (niedomiar), OE (nadmiar), ZE (dzielenie przez zero), DE (argument w postaci nieunormowanej, IE (operacja błędna)

- Rejestr adresu rozkazu, rejestr kodu rozkazu i rejestr adresu danej

1.19.3 Lista rozkazów

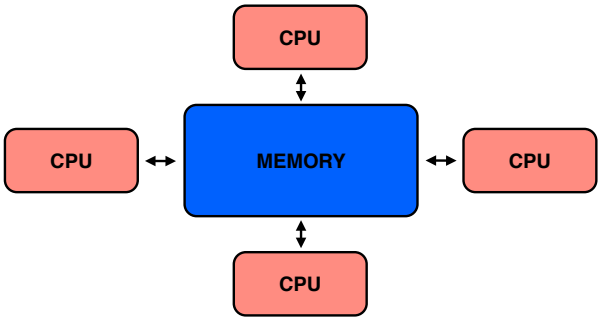
- **Rozkazy przesłań** przeznaczone do ładowania liczb z pamięci lub odsyłania ich do pamięci. Dla różnego rodzaju liczb używa się różnych rozkazów. Rozkazy z przyrostkiem **P** usuwają wysłaną liczbę ze stosu.
- **Rozkazy arytmetyczne** pozwalają wykonywać operacje dodawania, odejmowania, mnożenia, dzielenia oraz inne operacje. W przypadku operacji odejmowania i dzielenia, oprócz rozkazów realizujących operacje w zwykłej postaci, istnieją także rozkazy wykonujące te operacje dla argumentów zamienionych miejscami na stosie. Oczywiście mamy rozkazy usuwające argument ze stosu i pozostawiające go.
- **Rozkazy porównań i testowania** pozwalają dokonać porównania dwóch liczb albo uzyskać charakterystyczną informację o liczbie w celu wykonania skoku warunkowego.
- **Rozkazy realizujące funkcje przestępne**
- **Rozkazy ładowania stałych**

1.20 Architektury MIMD

MIMD, tj. **Multiple Instruction Multiple Data** to rodzaj architektury, w której przetwarzanie równoległe zachodzi zarówno na poziomie danych jak i instrukcji. Komputery zbudowane w architekturze MIMD posiadają wiele procesorów, zwykle SIMD, pracujących niezależnie i asynchronicznie. Mogą one korzystać ze wspólnej pamięci dzielonej lub używać modelu rozproszonego w którym każdy z nich posiada prywatną przestrzeń adresową. W ramach MIMD wyróżniamy trzy kategorie: MIMD–SM, MIMD–DM, MIMD–HDSM.

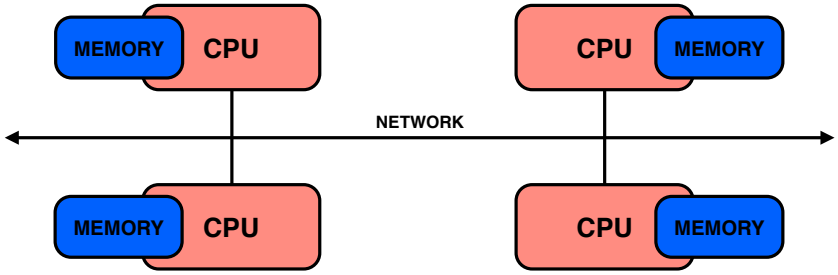
- **MIMD Shared Memory** – prcesory połączone specjalizowaną siecią *interconnect*, poprzez którą komunikują się ze wspólnym obszarem pamięci. W ramach systemów SM można wyróżnić dwie kategorie, różniące się podejściem do problemu współdzielenia zasobów:
  - *shared everything* – odczyt pamięci dzielonej, ale wszystkie dane programu przechowywane są we wspólnym obszarze pamięci
  - *shared something* – odczyt pamięci dzielonej, ale użytkownik musi wyraźnie zdefiniować, które dane powinny się w tym obszarze znaleźć
- **MIMD Distributed Memory** – w odróżnieniu od SM, pamięć nie jest współdzielona między procesorami (każdy procesor ma własny obszar pamięci), sieć *interconnect* służy do wymiany komunikatów pomiędzy procesorami
- **MIMD Hybrid Distributed–Shared Memory**

1.20.1 MIMD Shared Memory



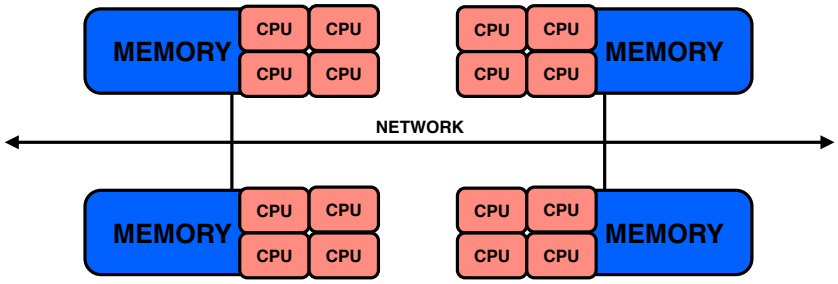
Rysunek IV: MIMD Shared Memory

1.20.2 MIMD Distributed Memory



Rysunek V: MIMD Distributed Memory

1.20.3 MIMD Hybrid Distributed–Shared Memory



Rysunek VI: MIMD Hybrid Distributed–Shared Memory

Gdyby ktoś się pogubił...

Spis treści

Zebrał i ułożył: Marcin Chwedziak (4 czerwca 2010)

Materiały pochodzą głównie ze slajdów dra. Mazurkiewicza, Google oraz http://fl9.eu/pwr/ptm/home (w tym przypadku to trochę pętla rekurencyjna)

Rysunek I pochodzi z Wikipedii. Pozostałe zostały stworzone przeze mnie.

LaTeX