Architektura komputerów.

Literatura:

1. Piotr Metzger, Anatomia PC, wyd. IX, Helion 2004
2. Scott Mueller, Rozbudowa i naprawa PC, wyd. XVIII, Helion 2009
3. Tomasz Kowalski, Urządzenia techniki komputerowej, Helion 2010
4. Paweł Bensel, Systemy i sieci komputerowe, Helion 2010

Komputer jest urządzeniem służącym do automatycznego przetwarzania danych. Jego istotną cechą jest możliwość programowania – funkcje, które realizuje zależą nie tylko od rozwiązań sprzętowych lecz głównie od programu, który wykonuje.

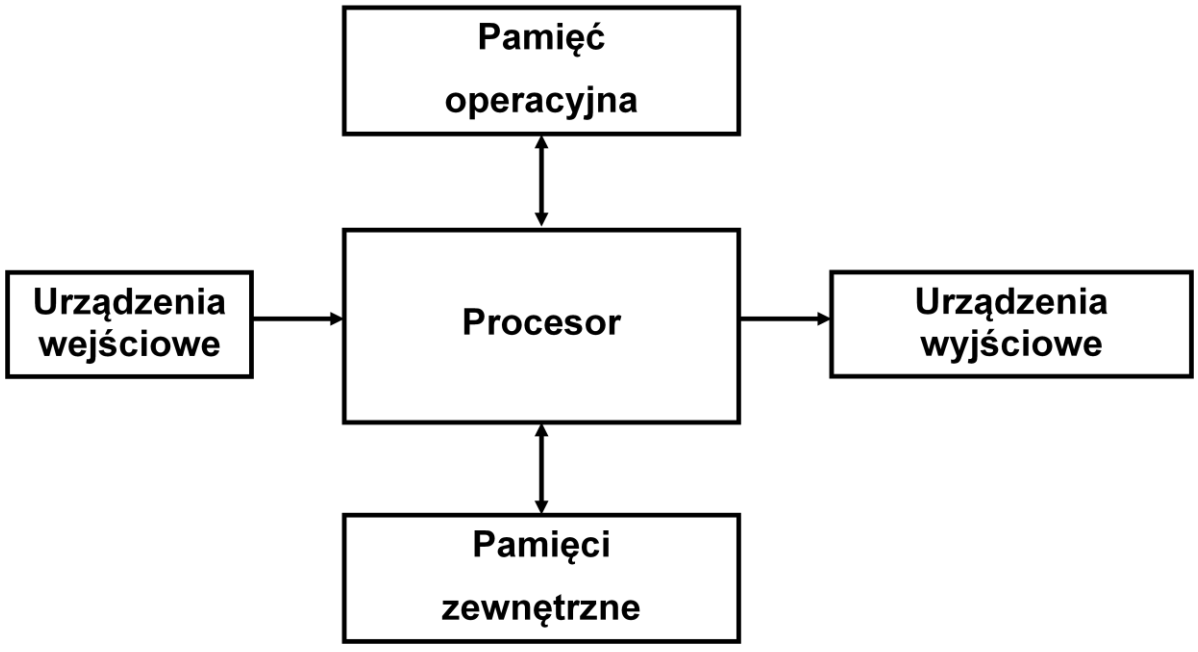
Komputery realizują następujące funkcje:

- Przetwarzanie danych (Data processing)
- Przechowywanie danych (Data storage)
- Przemieszczanie i duplikowanie danych (Data movement)
- Sterowanie (Control)

Architektura to atrybuty komputera widziane przez programistę np. zestaw instrukcji, bitowa reprezentacja danych, mechanizm wejścia-wyjścia, tryby adresowania. np. Czy dostępna jest instrukcja mnożenia ?

Organizacja komputera to sposób realizacji architektury, odnosi się do jednostek operacyjnych i ich połączeń Sygnały sterowania, interfejsy, technologia pamięci. np. Czy występuje dedykowana jednostka sprzętowa odpowiedzialna za operacje mnożenia, czy jest ona realizowana poprzez wielokrotne dodawanie?

Schemat blokowy komputera



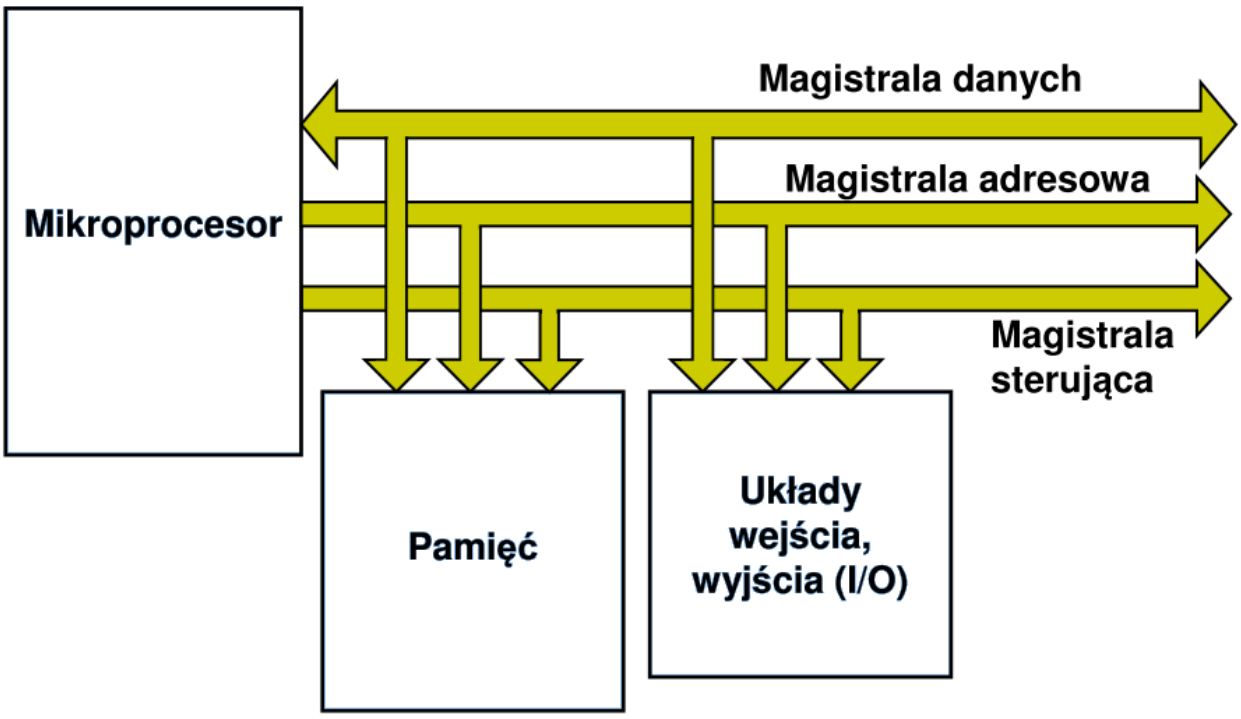
Procesor – układ cyfrowy przetwarzający dane zgodnie z programem zapisanym w pamięci komputera w postaci ciągu instrukcji.
Mikroprocesor – procesor zintegrowany w jednym układzie scalonym wielkiej skali integracji.

Ze względu na **sposób organizacji pamięci** i wykonywania programu:

- **architektura von Neumanna** – zarówno dane, jak i kod programu przechowywany jest w tym samym obszarze pamięci;
- **architektura harwardzka** – rozkazy i dane przechowywane są w odseparowanych obszarach pamięci;
- architektura mieszana – połączenie dwóch powyższych typów: obszary pamięci dla rozkazów i danych są odseparowane, jednak wykorzystują wspólne magistrale.

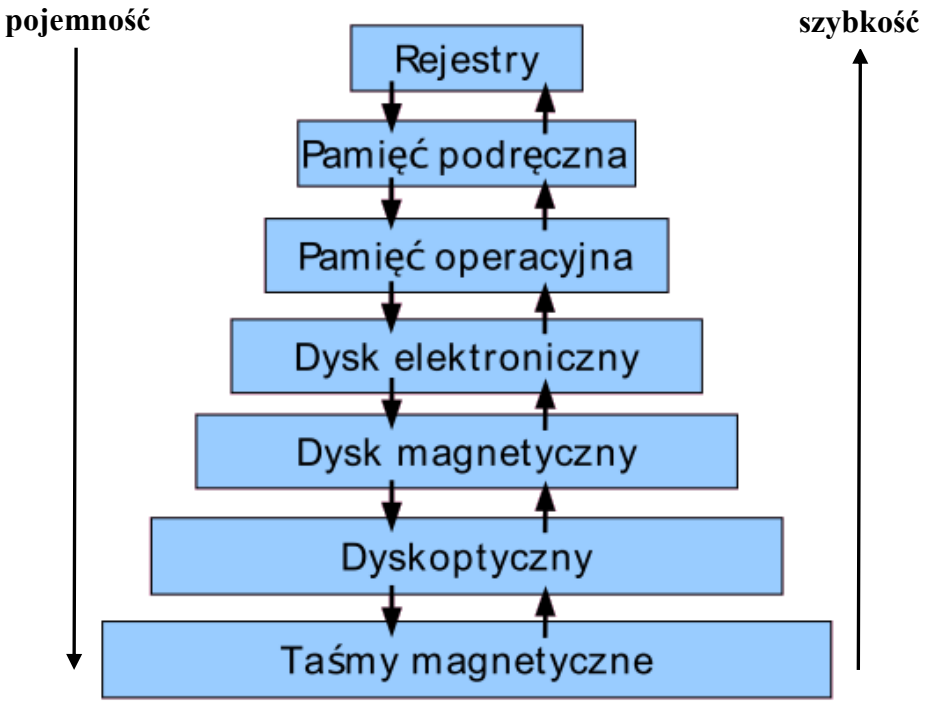
Instrukcje tworzące program są przechowywane w pamięci w taki sam sposób jak dane (komputer z zapamiętanym programem) – **koncepcja Johna von Neumanna**
Pamięć składa się z pewnej liczby ponumerowanych komórek, dostęp do pamięci następuje poprzez podanie przez procesor numeru komórki – jej adresu.
Komputer będzie pobierał kolejne instrukcje programu z kolejnych komórek pamięci (wykonywanie sekwencyjne programu) z wyjątkiem trzech sytuacji:
- wykonanie instrukcji skoku (lub pętli),
- wywołania podprogramu (procedury) i powrotu z niej,
- zgłoszenia przerwania i powrotu po obsłudze tego zdarzenia.

Adres komórki z której pobierany będzie następny rozkaz przechowywany jest w specjalnym rejestrze procesora – liczniku rozkazów.

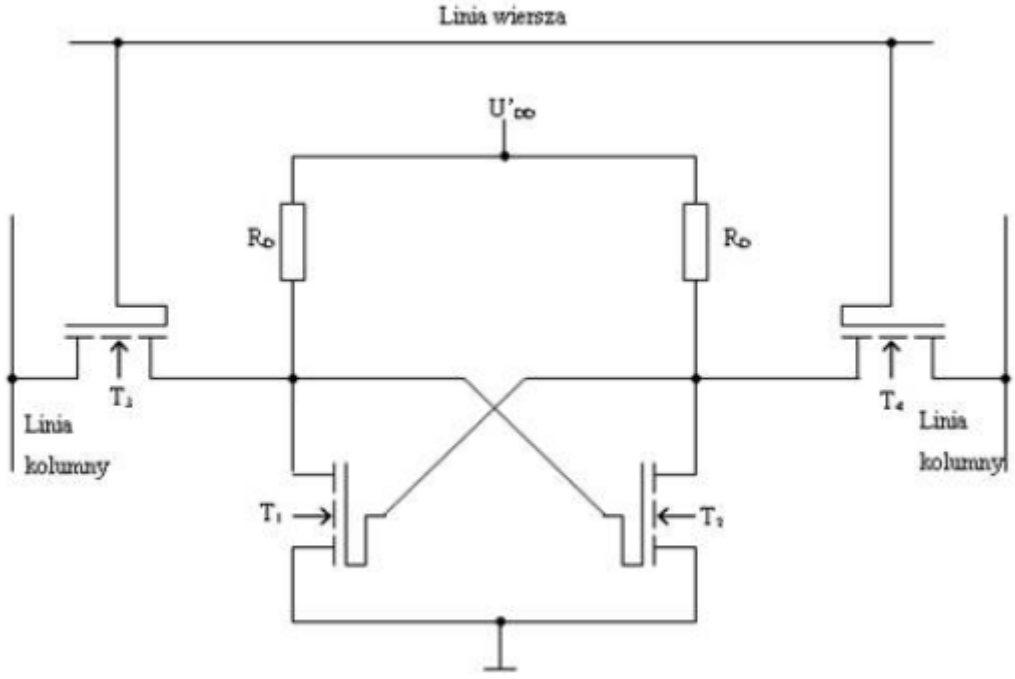


Schemat blokowy mikrokomputera

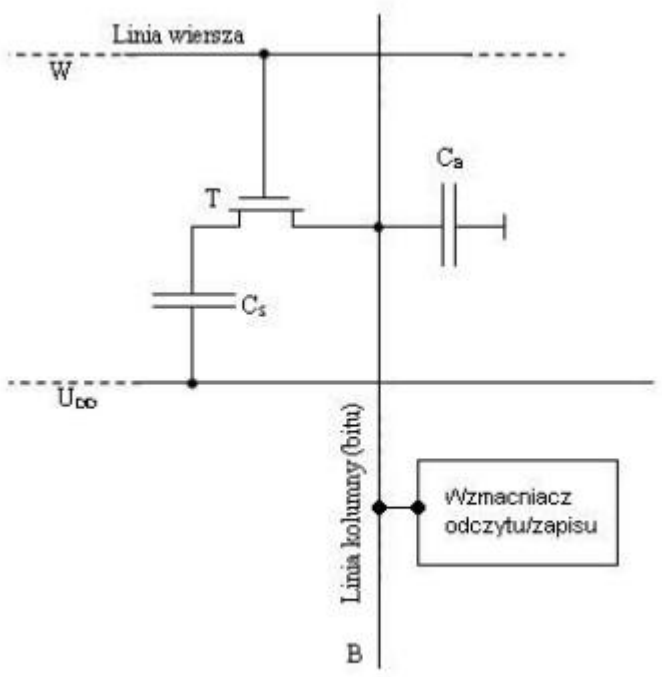
Pamięć komputera powinna mieć dużą pojemność, dużą szybkość, niski koszt, przechowywać dane i programy w sposób trwały. Ze względu na sprzeczność tych wymagań w systemach komputerowych stosuje się hierarchiczny system pamięci.



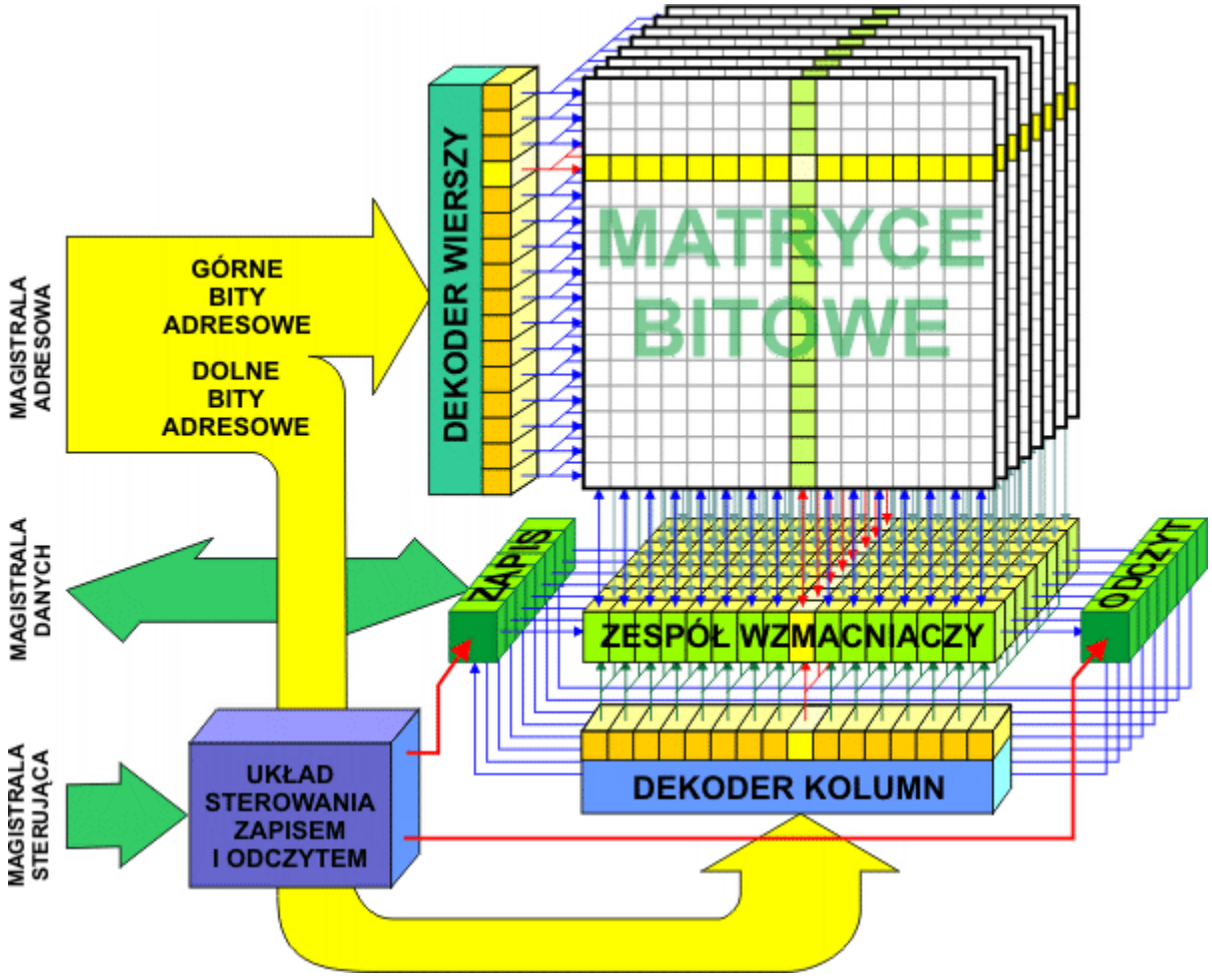
Hierarchia pamięci w systemie komputerowym



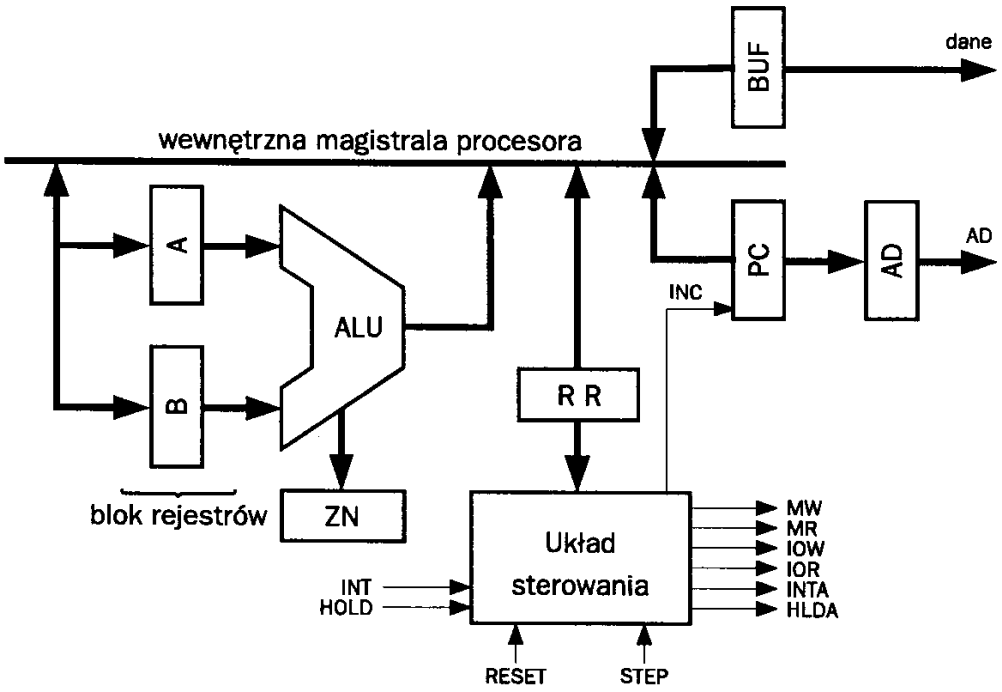
Komórka pamięci statycznej

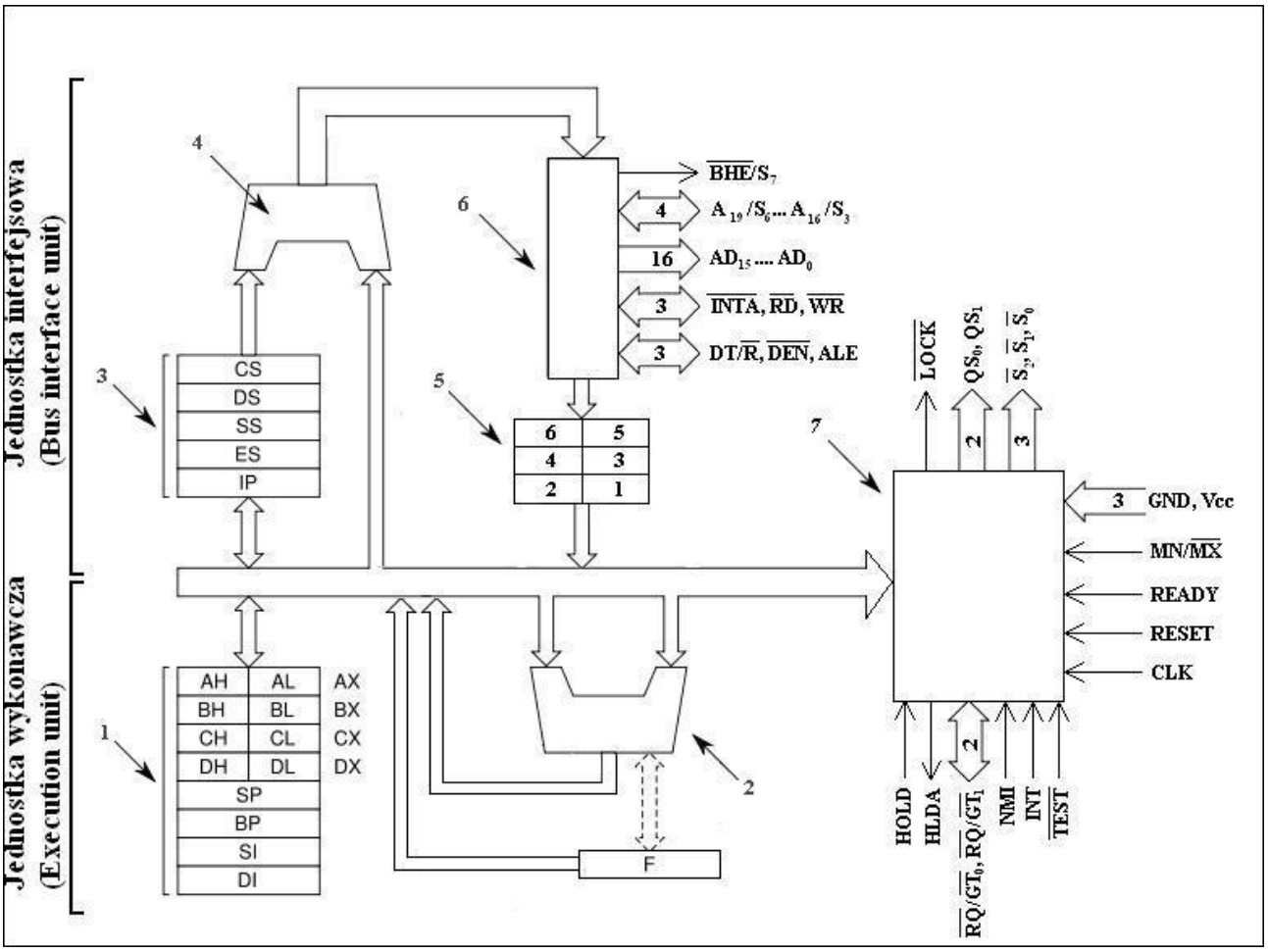


Komórka pamięci dynamicznej

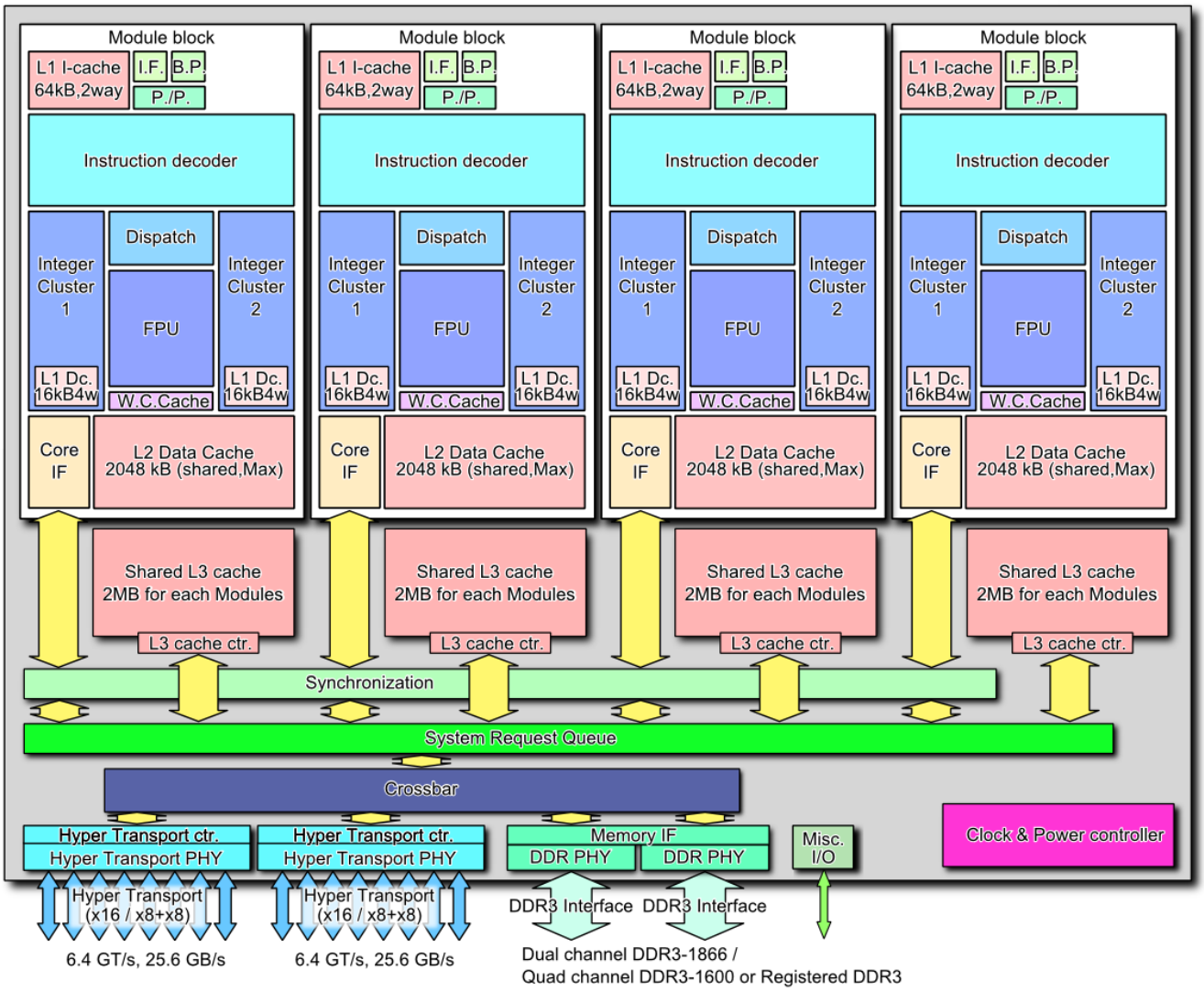


Organizacja typowego mikroprocesora





Architektura procesora 8086



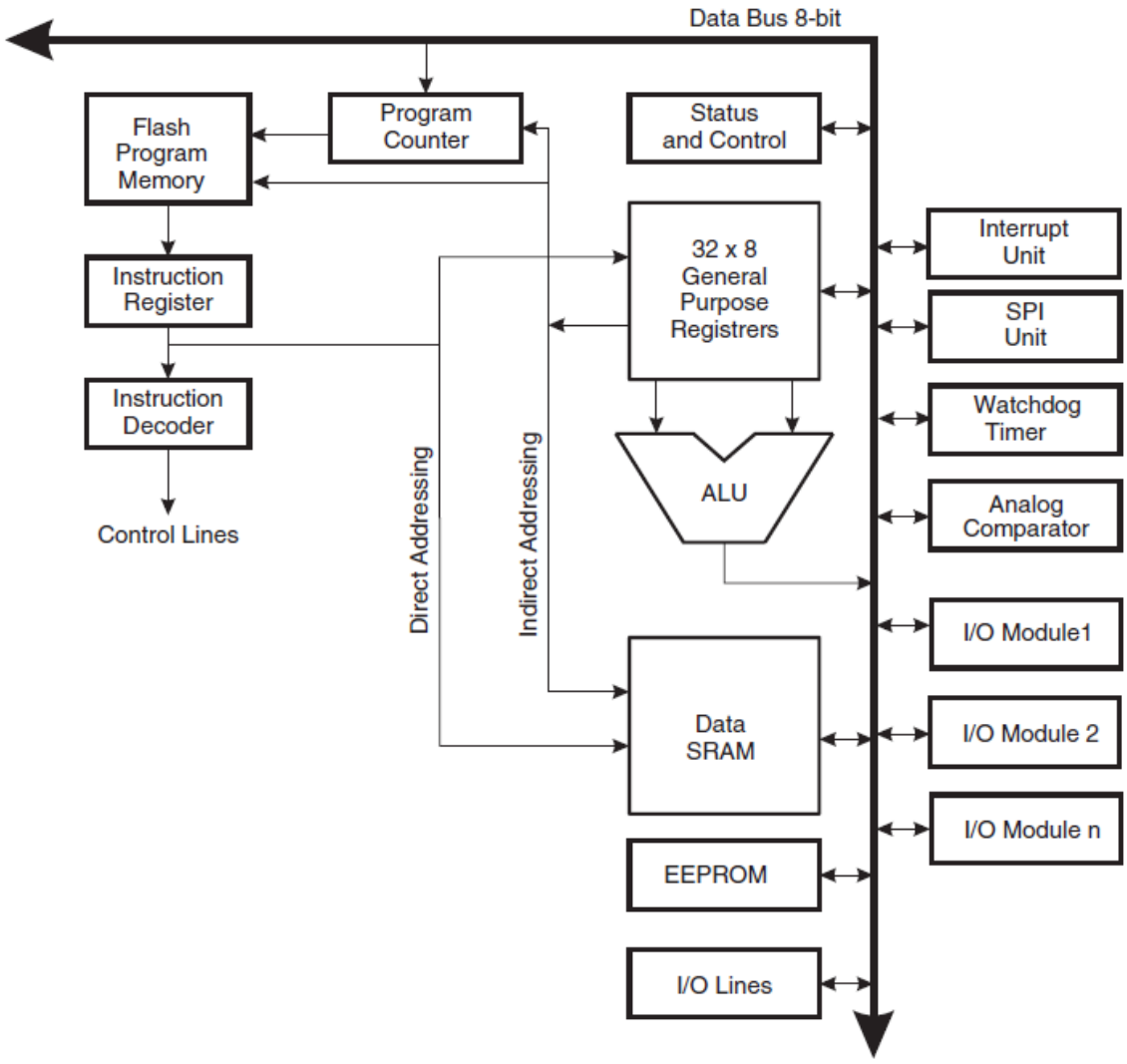
Architektura współczesnego procesora

Instrukcja i mikrooperacja

Instrukcja – najmniejsza jednostka programu, kod binarny określający sekwencję mikrooperacji

Mikrooperacja – zbiór wszystkich operacji wykonywanych przez mikroprocesor w pojedynczej fazie cyklu rozkazowego

Lista instrukcji – zbiór wszystkich instrukcji wykonywanych przez dany mikroprocesor
Instrukcje są zewnętrzne w stosunku do mikroprocesora. Ciąg instrukcji w pamięci komputera, realizujący wybrany **algorytm obliczeniowy** jest nazywany **programem**.



Schemat blokowy procesora AVR Mega o architekturze harwardzkiej

Wszystkie mikroprocesory zawierają podobne elementy:

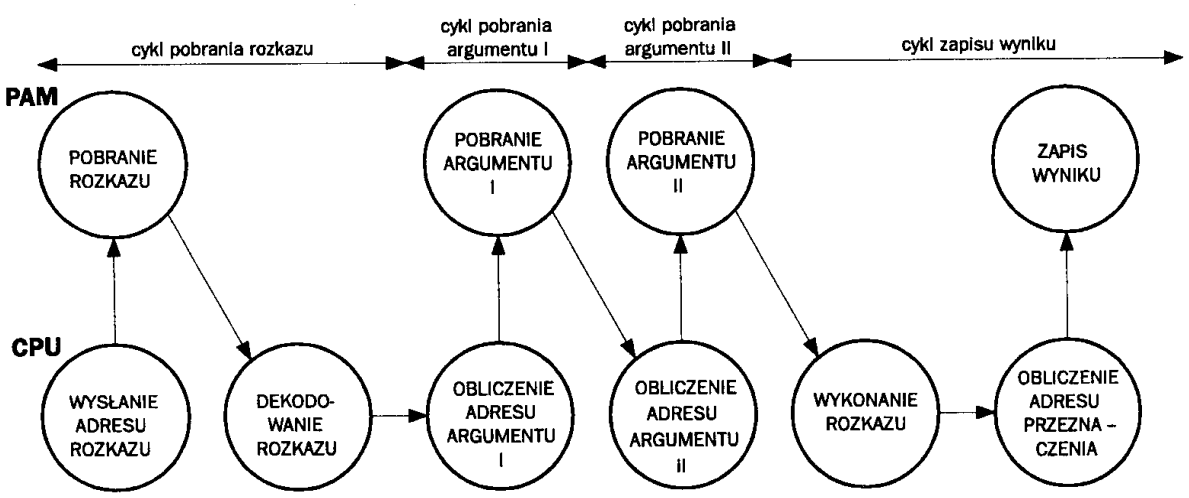
- układ sterowania i synchronizacji, który kontroluje pracę procesora i wytwarza sygnały potrzebne do sterowania poszczególnymi elementami komputera
- arytmometr, czyli układ, który wykonuje operacje arytmetyczne i logiczne
- rejestry, tj. układy pamięci
- wewnętrzne szyny łączące elementy procesora

Podstawowymi rejestrami, które znajdują się w każdym mikroprocesorze, są:

- **licznik** rozkazów - zawiera on adres następnego rozkazu do wykonania. Podczas cyklu rozkazowego automatycznie zwiększany o długość bieżącego rozkazu, chyba że ostatnim wykonywanym rozkazem był efektywny skok, wywołano podprogram lub przyjęto przerwanie.
- **rejestr rozkazów** - zawiera kod aktualnie wykonywanego rozkazu
- **akumulator**, jest używany w czasie wykonywania rozkazów arytmetycznych, logicznych, I/O i in.; niektóre procesory mają kilka takich rejestrów
- **rejestr znaczników** - zawiera dodatkowe informacje o wyniku operacji arytmetyczno-logicznych, np. "wynik równy zeru"

W procesorze układ sterowania działa cyklicznie, wykonując cykl rozkazowy. **Cykl rozkazowy** składa się z dwóch faz

- Fazy pobrania rozkazów.
- Fazy wykonania rozkazów.



W fazie pobrania rozkazu na magistralę adresową wysyłana jest zawartość licznika rozkazów. Licznik rozkazów zawiera adres komórki pamięci zawierającej rozkaz, który ma być w danej chwili wykonany. Po odczytaniu z pamięci rozkaz wędruje magistralą danych do procesora i wpisuje się do rejestru rozkazów. Na końcu fazy pobrania rozkazów układ sterowania zwiększa zawartość licznika o liczbę pobranych bajtów.

W fazie wykonania rozkazów układ sterowania odczytuje z rejestru rozkazów rozkaz, dokonuje jego dekodowania i w zależności od rodzajów rozkazów generuje odpowiednie sygnały sterujące. We współczesnych procesorach oba te cykle wykonywane są jednocześnie. W czasie wykonywania rozkazu pobierany jest już następny. Zbiór wszystkich możliwych do wykonania przez procesor rozkazów nazywamy **listą rozkazów**.
Rozkazy te podzielone są na cztery grupy:

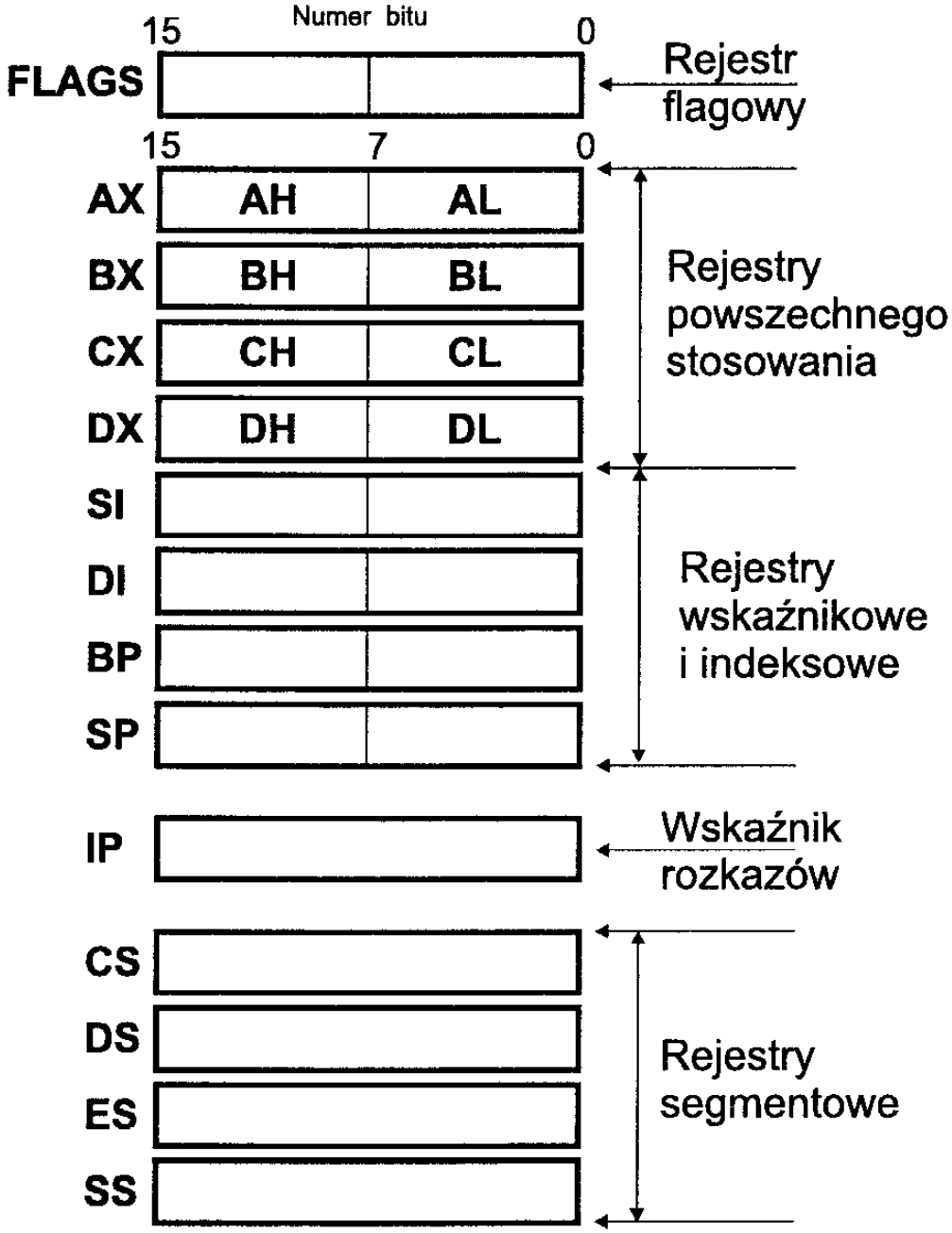
- służące do przesyłania informacji
- arytmetyczne i logiczne
- sterujące wykonaniem programu (rozkazy skoków)
- wejścia-wyjścia

Na cykl rozkazowy składają się jeden lub kilka cykli maszynowych.

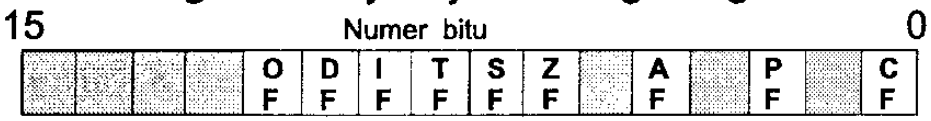
Cykl maszynowy, to cykl, w którym następuje przesłanie danych (odczyt lub zapis) między jednostką centralną a pamięciową lub układem wejścia – wyjścia, w zależności od rodzaju przesłania rozróżnia się cykl maszynowy: pobrania kodu operacji, odczytu i zapisu pamięci, odczytu i zapisu wejścia - wyjścia, przyjęcia przerwania itd. W każdym cyklu maszynowym następuje wysłanie:
- adresu na magistralę adresową,
- danych na magistralę danych,
- sygnałów sterujących, informujących o rodzaju cyklu, na magistralę sterującą.

Układy pamięci lub wejścia - wyjścia powinny w tym czasie wykonać odpowiednie czynności - zapisać dane lub wysłać je na magistralę danych. Jeden cykl maszynowy wykonywany jest w czasie jednego lub kilku (w zależności od procesora i rodzaju cyklu) taktów zegara.

Rejestry procesora 8086

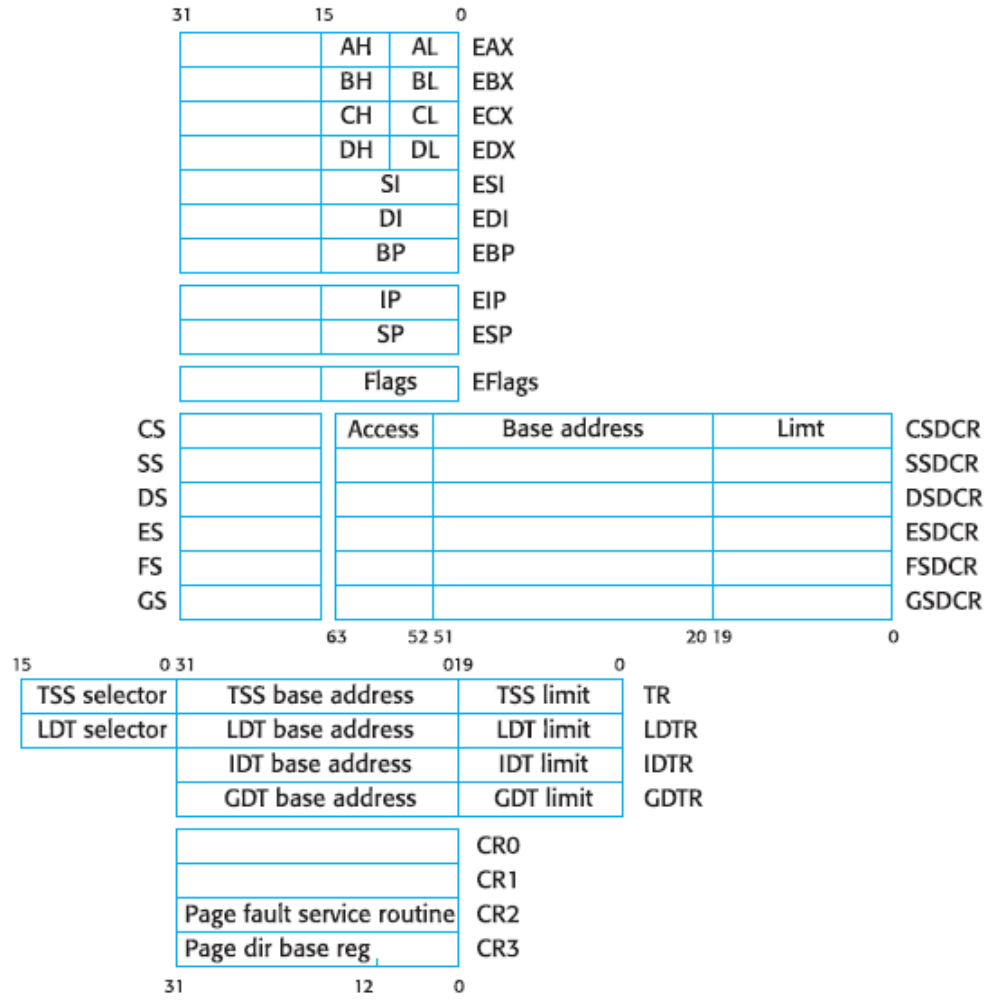


Poszczególne bity rejestru flagowego



OF - flaga nadmiaru (ang. *overflow flag*)
DF - flaga kierunku (ang. *direction flag*)
IF - flaga zezwolenia na przerwanie (ang. *Interupt enable flag*)
TF - flaga pracy krokowej (ang. *trap flag*)
SF - flaga znaku (ang. *sign flag*)
ZF - flaga zera (ang. *zero flag*)
AF - flaga przeniesienia pomocniczego (ang. *auxiliary flag*)
PF - flaga parzystości (ang. *parity flag*)

Fig. 7.5
Intel 80x86/Pentium
CPU register set.



ID Identification flag or CPUID availability
VIP Virtual interrupt pending
VI Virtual interrupt active
AC Alignment check
VM Virtual 8086 mode active
RFR Resume task after breakpoint interrupt
NT Nested task
IOPL IO privilege level
O Arithmetic overflow error
D Direction of accessing string arrays
IE External interrupt enable
T Trap, single-step debugging, generates an INT #1 after each instruction
S Sign, MS bit value
Z Zero, result being zero
A Auxiliary carry, used by BCD arithmetic on 4 LS bits
P Parity, operand status
C Carry, indicates an arithmetic carry or borrow result

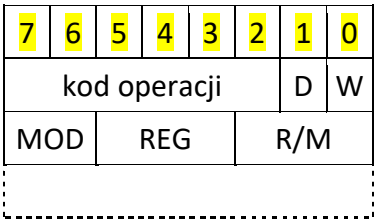
Fig. 7.9
CPU status flags.

Kodowanie rozkazów

kod operacji	adres 1 argumentu	adres lub wartość 2 argumentu

Struktura kodu rozkazu dwuargumentowego

Rozkazy mikroprocesora 8086 są wielobajtowe. Liczba bajtów każdego rozkazu zależy od jego rodzaju i może wynosić od jednego do sześciu.



Struktura kodu rozkazu procesora 8086

Pierwszy bajt zawiera sześciobitowy kod operacji oraz dwa bity (kierunku i szerokości). Bit *D* określa kierunek transmisji (*0* - wynik operacji jest przesyłany z rejestru do pamięci, *1* - z pamięci do rejestru). W zależności od wartości tego bitu w rozkazie rozróżniane są operandy źródłowe i operandy przeznaczenia. Bit *W* określa szerokość operandu danego rozkazu (0 - operacje bajtowe, 1 - operacje na słowie 16-bitowym).

Jeżeli rozkaz jest wielobajtowy, to drugi bajt rozkazu określa sposób adresowania argumentów. Zawiera on trzy grupy bitów

- dwubitowa grupa *MOD* określa tryb adresowania
- trzybitowa grupa *REG* określa numer rejestru, w którym znajduje się operand
- trzybitowa grupa *R/M* określa sposób wyznaczenia miejsca operandu

Jeżeli operandy znajdują się w rejestrach mikroprocesora (*MOD* = 11), to pola *REG* i *R/M* stanowią ich numery (odpowiednio pierwszego i drugiego operandu)

REG R/M	W = 0	W = 1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

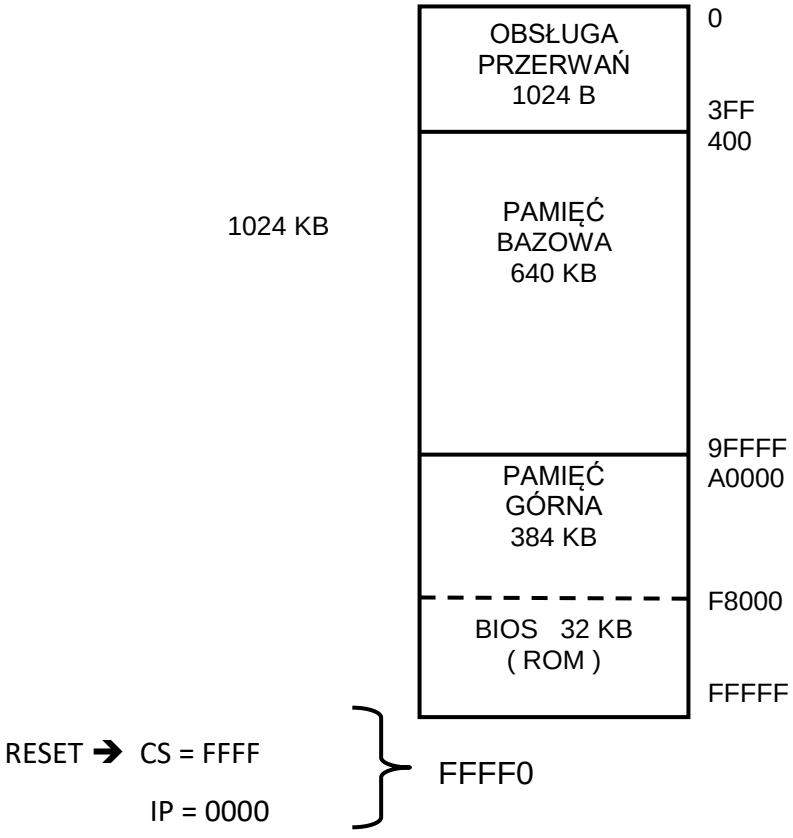
Jeżeli jeden z operandów znajduje się w pamięci, to pola *MOD* i *R/M* określają jego adres

R/M	MOD = 00	MOD = 01	MOD = 10
000	BX+SI	BX+SI+p8	BX+SI+p16
001	BX+DI	BX+DI+p8	BX+DI+p16
010	BP+SI	BP+SI+p8	BP+SI+p16
011	BP+DI	BP+DI+p8	BP+DI+p16
100	SI	SI+p8	SI+p16
101	DI	DI+p8	DI+p16
110	p16	BP+p8	BP+p16
111	BX	BX+p8	BX+p16

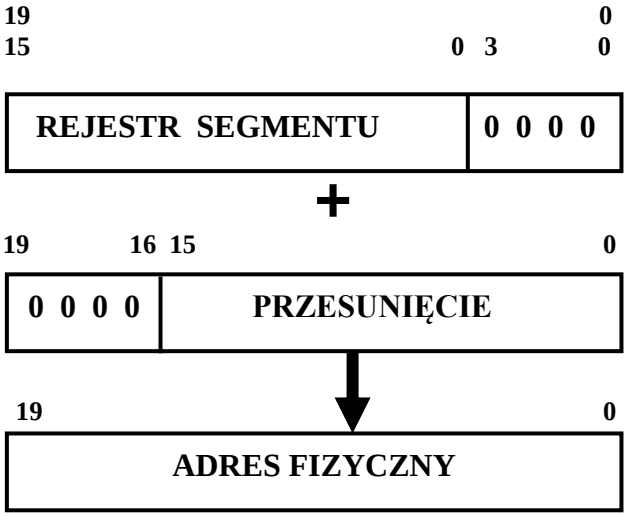
Znaczenie bitów *R/M* zależy od wartości bitów w polu MOD. Jeżeli *MOD* != 11, to grupa *R/M* określa rejestry adresujące. Jeżeli natomiast *MOD* == 11, to *R/M* określa rejestr (podobnie jak *REG*) drugiego operandu. Jeżeli rozkaz tego wymaga, to po drugim bajcie rozkazu może występować jeden lub dwa bajty przemieszczenia. Jeden bajt przemieszczenia występuje w sytuacji, gdy *MOD* == 01, natomiast przemieszczenie dwubajtowe występuje, gdy *MOD* == 10.

Organizacja pamięci operacyjnej

Wyróżnione obszary Pamięci Operacyjnej



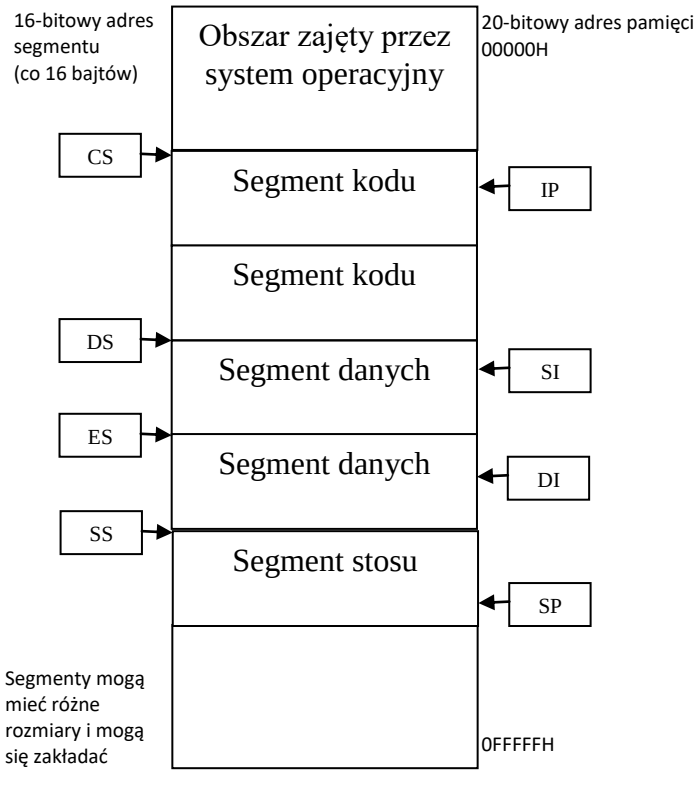
Adresowanie pamięci operacyjnej, segmentacja



- rejestr segmentu : CS, DS, SS, ES
- przesunięcie : IP, SP, ARG (BX, BP, SI, DI)
- początki segmentów co 16 bajtów
- max długość segmentu : 64 kB
- max wielkość pamięci : 1 MB

Wykorzystanie rejestrów segmentów :

- CS ➔ segment rozkazów, przesunięcie z IP
- DS ➔ segment danych, przesunięcie z ARG
- SS ➔ segment stosu, przesunięcie z ARG (SP)
- ES ➔ segment danych dodatkowych, przesunięcie z ARG



Instrukcje przesyłania danych

Podstawową instrukcją przesyłania danych jest instrukcja **mov dst, src**

Pozwala ona przesłać do miejsca przeznaczenia (rejestru, komórki pamięci) zawartość ze źródła (rejestru, komórki pamięci lub wartość liczbowa). Istnieją przy tym pewne ograniczenia, np. nie można wpisać bezpośrednio wartości liczbowej do rejestru segmentowego, dlatego używamy dwóch instrukcji:

```
mov ax, data
mov ds, ax
```

Nie można wpisać wartości do licznika rozkazów (rejestru CS:IP) – służą do tego rozkazy skoku lub wywołania podprogramu.

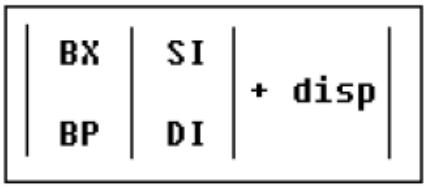
Przykłady wykorzystania rozkazu ***mov dst, src***

```
mov   AX, 0B00h
mov   DS, AX
mov   CL, 'A'
mov   CH, 01101001b
mov   BX, 15Eh
mov   [BX], CX
```

Tryby adresowania

Trybem adresowania nazywamy sposób wyznaczania adresu operandu, którego to mianem określamy argumenty i wyniki operacji. Adres operandu obliczany jest zgodnie z równaniem

EA = BR + IR + p



gdzie:

- ***EA*** - adres efektywny
- ***BR*** - rejestr bazowy
- ***IR*** - rejestr indeksowy
- ***p*** - przemieszczenie

Adres efektywny *EA* jest adresem logicznym "widzianym" przez program. Na podstawie *EA* układy segmentacji obliczają adres rzeczywisty w pamięci operacyjnej. Rejestrem bazowym może być rejestr *BP* lub *BX*, a rejestrem indeksowym może być rejestr *SI* lub *DI*. Przemieszczenie jest zawarte w rozkazie i może mieć długość ośmiu lub szesnastu bitów.

Mikroprocesor 8086 realizuje następujące tryby adresowania:

- natychmiastowe
- rejestrowe
- bezpośrednie
- pośrednie
- bazowe
- indeksowe
- indeksowo-bazowe
- adresowanie względne

Adresowanie natychmiastowe

W adresowaniu natychmiastowym argument pobierany jest bezpośrednio z rozkazu. np. **mov AX, 100** – w rejestrze *AX* zostanie zapisana liczba 100.

Adresowanie rejestrowe

W adresowaniu rejestrowym operandy znajdują się w rejestrach wewnętrznych mikroprocesora. np. **mov AX, BX** – w rejestrze *AX* zostanie zapisana zawartość rejestru *BX*.

Adresowanie bezpośrednie

W adresowaniu bezpośrednim adres operandu znajduje się bezpośrednio w rozkazie. np. **mov AX, [100]** – w rejestrze *AX* zostanie zapisana zawartość komórki pamięci (segment danych) o adresie 100.

Standardowy adres operandu jest przesunięciem w segmencie danych (*DS*), można to nadpisać poprzez wskazanie innego segmentu. Np. mov AX, CS:[40] – w rejestrze *AX* zostanie zapisana zawartość z komórki pamięci (segment programu(kodu)) o offsecie 40.

Adresowanie pośrednie

W trybie adresowania pośredniego w rozkazie znajduje się nie adres argumentu lecz adres miejsca, gdzie znajduje się argument operacji. Np. **mov AX, [BX]** – w rejestrze AX zostanie zapisana zawartość komórki pamięci o adresie, który znajduje się w rejestrze *BX*.

Wszystkie rejestry wskazują offset w segmencie danych (*DS*), poza rejestrem *BP*, który jest przesunięciem w segmencie stosu (*SS*). Można to zmienić określając segment w rozkazie.

np. mov AX, SS:[BX] – w rejestrze *AX* zostanie zapisana zawartość komórki pamięci z segmentu stosu o adresie, który znajduje się w rejestrze *BX*.

Pobieranie adresu zmiennej

Istnieje możliwość pobrania adresu zmiennej za pomocą instrukcji **LEA** (Load Effective Address) lub operatora **OFFSET**. Jest to wykorzystywane przy adresowaniu pośrednim oraz przy przekazywaniu parametrów do procedur.

Przykład:

```
org  100h
mov  AL, zmX
lea   BX, zmX
mov  byte ptr [BX], 55h
mov  AL, zmX
ret

zmX  DB  22h
end
```

lub z operatorem OFFSET

```
org 100h
mov  AL, zmX
mov  BX, offset  zmX
mov  byte ptr [BX], 55h
mov  AL, zmX
ret

zmX  DB  22h
end
```

Instrukcja LEA jest bardziej elastyczna niż operator OFFSET gdyż pozwala pobrać adres zmiennej z indeksem np. lea BX, tablica[4]

Adresowanie bazowe

Adresowanie bazowe jest to rodzaj adresowania pośredniego, gdzie rozkaz wskazuje na jeden z rejestrów bazowych *BX* lub *BP* i może zawierać 8- lub 16-bitową wartość stanowiącą lokalne przemieszczenie. Adresem efektywnym jest suma zawartości rejestru bazowego i lokalnego przemieszczenia. Np. **mov AX, [BP+2]**.

Adresowanie indeksowe

Adresowanie indeksowe jest rodzajem adresowania pośredniego, gdzie adres efektywny jest sumą zawartości rejestru indeksowego *SI* lub *DI* i lokalnego przemieszczenia. Np. **mov AX, [SI+3].**

Adresowanie bazowo-indeksowe

W adresowaniu bazowo-indeksowym, adres efektywny jest sumą zawartości jednego z rejestrów bazowych, jednego z rejestrów indeksowych i lokalnego przemieszczenia. Np. **mov AX, [SI+BP+4].**

Adresowanie względne

Jest stosowane w skokach – służy do przeniesienia sterowania o pewną liczbę pozycji względem aktualnie wykonywanej instrukcji.

Rozkazy operujące na ciągach słów

Rozkazy operujące na ciągach słów posługują się rejestrami indeksowymi. Rejestry *SI* i *DI* zawierają adresy efektywne pierwszego słowa odpowiednio w ciągu źródłowym i wynikowym. Po każdej transmisji rejestry indeksowe są automatycznie inkrementowane lub dekrementowane w zależności od ustawienia bitu *DF* w rejestrze znaczników.

Rozkazy operujące na rejestrach WE/WY

Rozkazy operujące na rejestrach WE/WY (**in** oraz **out**) zawierają adres WE/WY (adres natychmiastowy) lub posługują się zawartością rejestru *DX* (adresowanie pośrednie).