Systemy komputerowe (W04)

Cykl rozkazu

A.Pruszkowski

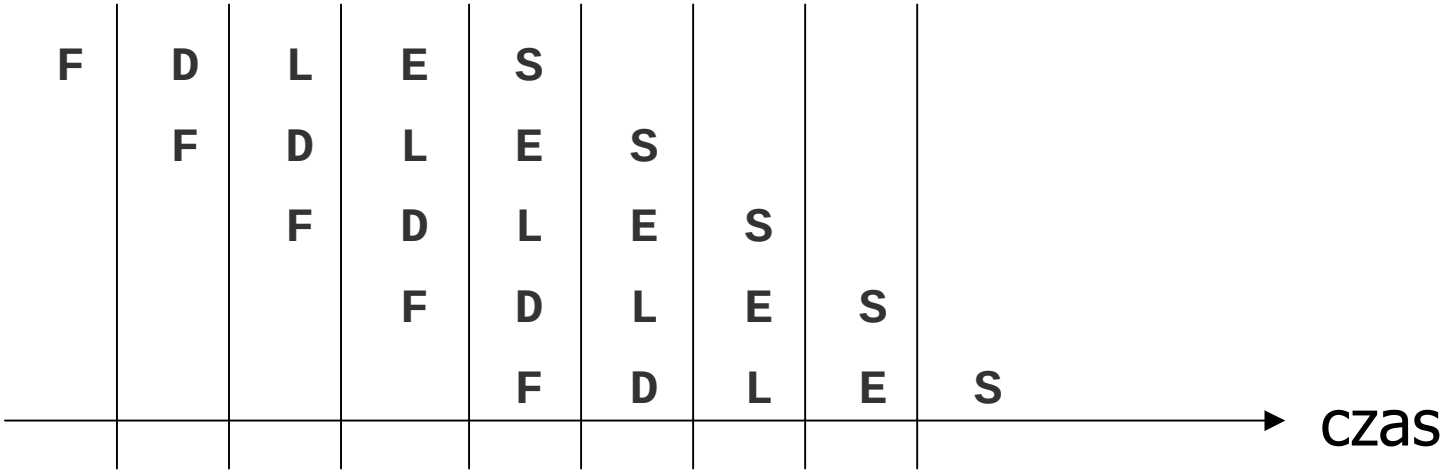
Fazy cyklu rozkazu w CPU

Systemy Komputerowe

- **Podstawowe fazy cyklu rozkazu w CPU**
  - **F: Pobranie rozkazu z pamięci (ang. fetch)**
    - z miejsca wskazanego w pamięci kodu przez PC
  - **D: Dekodowanie**
  - **L: Pobranie argumentów**
    - pobranie adresu pośredniego z pamięci - faza opcjonalna
      - gdy instrukcja wykonuje się w trybie adresowania pośredniego pamięci
  - **E: Wykonanie**
    - tę fazę wykonuje głównie ALU
  - **S: Zapisanie wyników**
  - **I: Obsługa przerwań**
    - opcjonalna
  - Ustalenie nowego PC może być część końcową F, E lub S - zależnie od przyjętej konwencji
    - w fazie F - przy skokach wartość może być liczona daremnie, ...
    - często w specyfikacji zapisuje się kiedy następuje zmiana PC (jest to np.: niezbędne aby wyznaczyć argument dla instrukcji skoku relatywnego względem PC)
  - Jedna faza zajmuje z reguły jeden takt zegara - inna nazwa fazy: mikrooperacja

- **Co można przyspieszyć w cyklu rozkazowym**
  - uprościć pobieranie argumentów i zapisywanie wyników
  - co jeszcze ???
- **A co robi ALU przez cały czas wykonywania rozkazu?**
  - Mamy fazy (pomińmy fazę I):

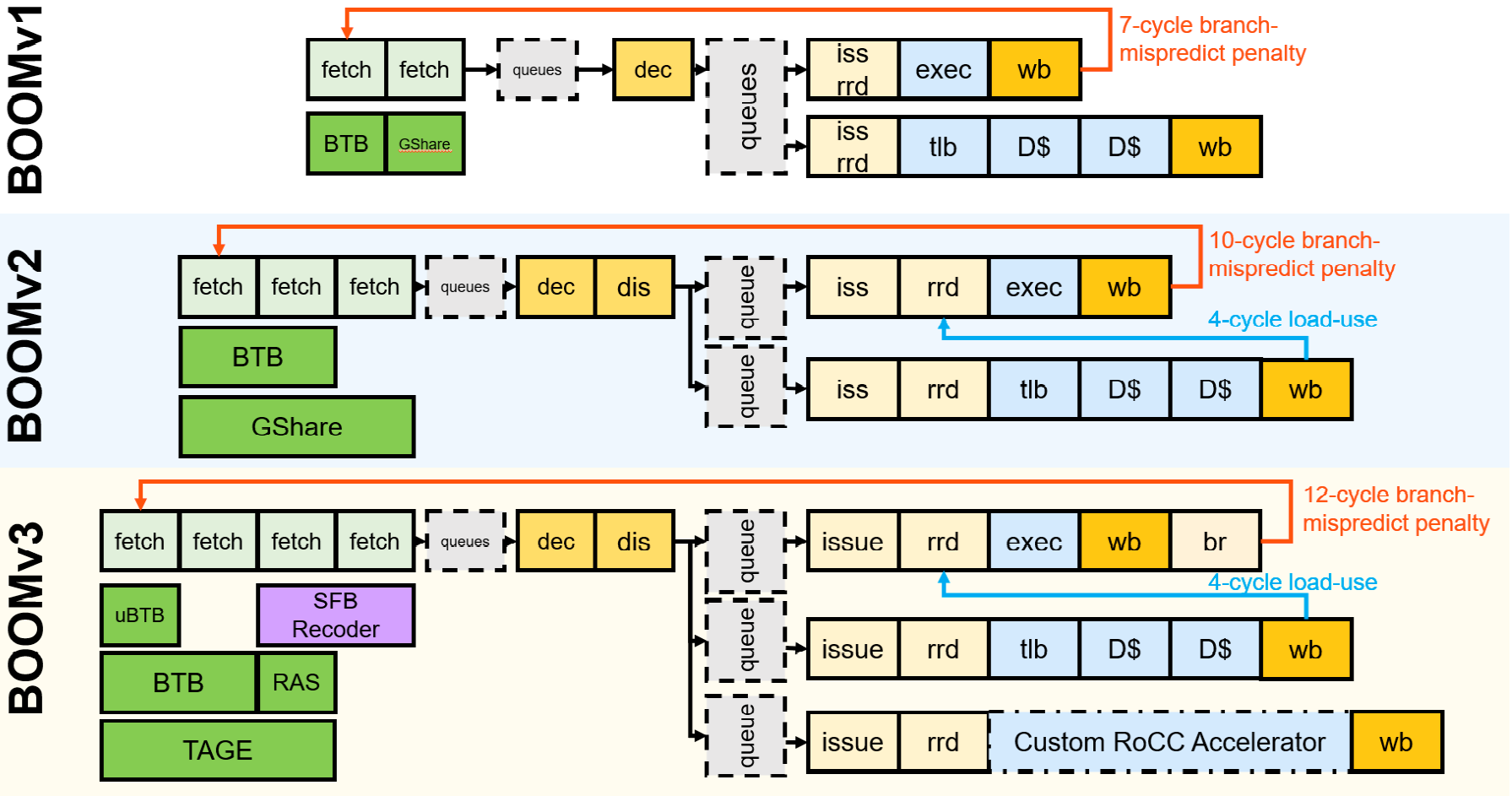
    F D L E S
  - Czyli 80% czasu ALU nie jest wykorzystywany!
    - podobnie inne elementy
  - Jak to zmienić - wprowadzając potoki



  - Tutaj CPU wykonuje każdą z faz non-stop!

Systemy Komputerowe

- **Wiele konstrukcji z przetwarzaniem potokowym ewoluowało**
  - Dla przykładu Berkeley Out-of-Order Machine (BOOM) - otwarta, zaopatrzona w możliwość ustalania wewnętrznych parametrów implementacja zgodna z RV64GC procesora RISC-V w języku Chisel



Zródło: https://github.com/riscv-boom/riscv-boom

Systemy Komputerowe

- **Konsekwencje wprowadzenia potoków**
  - **Pozytywne**
    - **zwiększenie wydajności procesora: 1 rozkaz na cykl maszynowy**
  - **Negatywne**
    - **nie wszystkie instrukcje da się tak wykonywać**
      - **skoki zaburzają ten układ, ciągi np.:**

```
beq     //gdy skok ma się wykonać
addi    //od tej instrukcji
addi
addi
xori    //do tej, CPU zacznie wykonywać te instrukcje ale nie ukończy ich
          wykonywania, a wyniki wstępne odrzuci
```

- **Konsekwencje wprowadzenia potoków**
  - **Negatywne**
    - **nie wszystkie instrukcje da się tak wykonywać**
      - **argumenty użyte w instrukcji mogą zależeć od wyników instrukcji wcześniejszych**

```
addi x1, x2, x3
addi x4, x1, x5 //ta instrukcja musi czekać na wynik poprzedniej
```

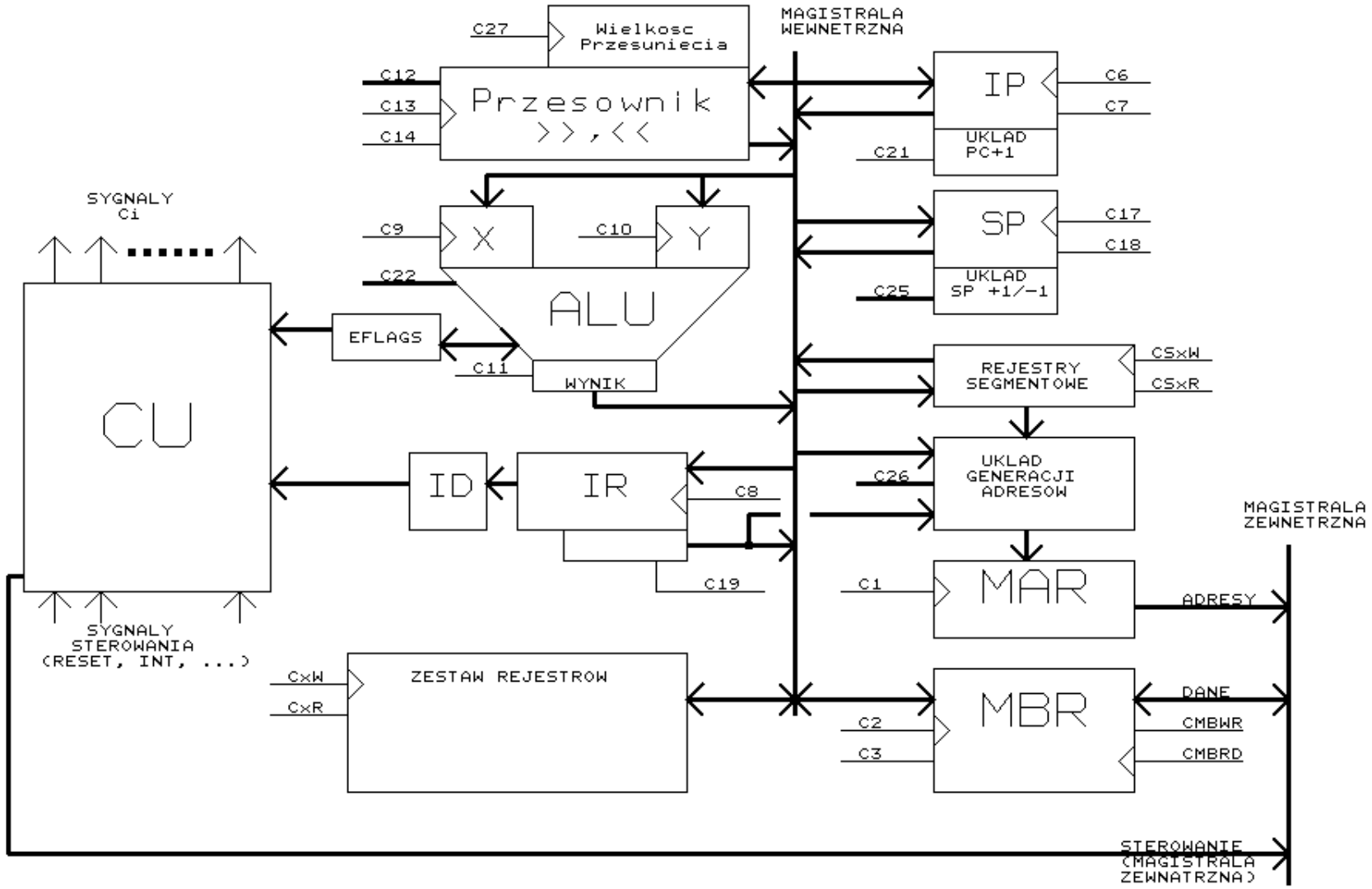
  - **Jak ten problem rozwiązać?**
    - **wstawić (co robi sam CPU) dwie instrukcje puste (np.: NOP)**

    - **co spowalnia procesor - nowoczesne kompilatory potrafią zreorganizować kod aby problem eliminować**

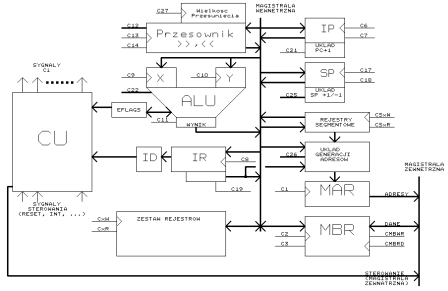
Budowa wewnętrzna CPU

- **Budowa wnętrza procesora**
  - **Istnieje bardzo dużo implementacji**
    - **dla każdej listy rozkazów (Instruction Set Architecture) można utworzyć wiele jej implementacji**
      - każda z nich ma swoje zalety i wady
      - faktyczna budowa CPU jest ściśle ukrywaną tajemnicą
        - detale w jej budowie sprawiają że CPU jest szybsze od konkurencyjnych rozwiązań
    - **dla przykładu mamy dziesiątki implementacji rdzeni Risc-V w różnych językach i różnych licencjach**
      - Rocket, Chisel, BSD
      - ReonV, VHDL, GPL v3
      - VexRiscv, SpinalHDL, MIT
      - RI5CY, SystemVerilog, Solderpad Hardware License (bliska Apache License, ale bez akceptacji fundacji)
      - ...
    - **istnieje równie dużo gotowych układów (System on a Chip) z rdzeniem Risc-V**
      - Rocket Chip, Rocket, BSD
      - PULPino, RI5CY, Solderpad Hardware License
      - ...

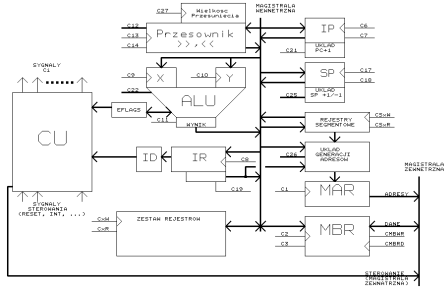
- **Konstrukcja hipotetycznego CPU**
  - nikt nie wykonał jego implementacji - ma jedynie pokazać problem dostępu do zasobów we wnętrzu CPU



Systemy Komputerowe



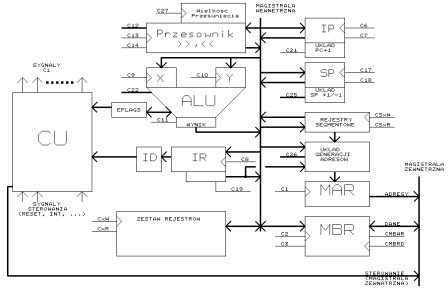
- **Hipotetyczny CPU**
  - Założenia:
    - istnieje tylko jedna magistrala komunikacyjna we wnętrzu CPU
      - tylko jeden transfer wewnątrz może mieć miejsce (!)
    - sygnały Cxx - to sygnały sterujące zachowaniem bloków
      - sygnały wyprowadzania danych na magistralę
        - C11 - wynik operacji wykonanej przez ALU
        - C19 - wyprowadzenie arg.wbudowanego czyli pola imm z rejestru opcode
        - CxR - wyprowadzenie zawartości jednego z rejestrów podanego w miejsce X (np.: A) na magistralę wewnętrzną
        - C3 - wyprowadzenie zawartości MBR na magistralę wew.
      - sygnały zatrzaskujące dane z magistrali (notacja jak z układów logicznych - trójkąt przy wejściu w dany blok)
        - C8 - zatrzaśnięcie danych w rejestrze opcode
        - C9 i C10 - zapisują na wejścia ALU odpowiednio X i Y dane z magistrali
        - CxW - zapisanie w rejestrze podanym w miejsce X (np.: A) informacji z magistrali wew.
        - C1 - zapisanie zawartości magistrali wew. w rejestrze MAR - jest on połączony z magistralą adresową połączoną z np.: pamięciami zewnętrznymi, C26 interpretacje można pominąć - steruje sposobem tworzenia adresu
        - C2 - zapis zawartości magistrali wew. w rejestrze MBR - jest on połączony z magistralą danych a ta z np.: pamięciami zewnętrznymi



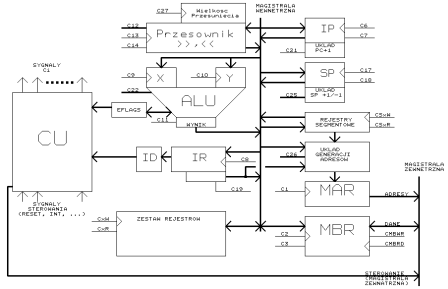
Hipotetyczny CPU

- Założenia:
  - sygnały Cxx - to sygnały sterujące zachowaniem bloków, np.:
    - sygnały sterujące pracą CPU
      - C21 - zwiększenie licznika rozkazów PC o 1
      - C22 - jest wielo bitowy (grubsza linia) ustala jaką operacje ma ALU zrealizować (np.: +, -, cmp, ...)
      - C25 - zwiększyć lub zmniejszyć zawartość SP o 1
    - pozostałe
      - CMBWR - uaktywnienie wyjścia rej. MBR - jego treść pojawia się na magistrali zewnętrznej
      - CMBRD - zapisanie zawartości zewnętrznej magistrali w rej. MBR (nie wpływa na zawartość mag. wewnętrznej)
  - jedna linia z zapisanymi sygnałami Cxx wykonywana jest w jednym cyklu zegarowym
    - kolejność sygnałów zapisana w jednej linii - dowolna, tylko dla wygody zapisujemy najpierw sygnały uaktywniające wyjścia a potem sygnały zatrzaskujące
      - CAR, C9 ≡ C9, CAR
      - ale łatwiej zrozumieć działanie zapisując w formacie źródło-cel/cele CAR, C9
    - finalnie sygnały sterujące zapisuje się w pamięci ROM jednostki sterującej CU

Systemy Komputerowe

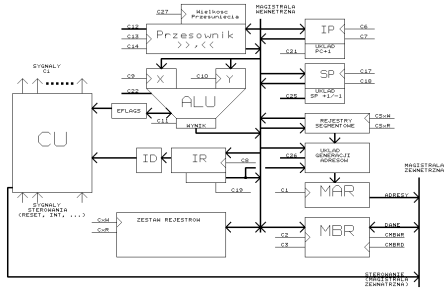


- **Hipotetyczny CPU**
  - Założenia:
    - konsekwencje posiadania jednej magistrali wewnętrznej
      - możemy tylko użyć jednego sygnału wyprowadzającego dane w określonym cyklu zegara
        - tj. C11, C19, C3, C14, C7, C18, CxR
      - z drugiej strony łatwo kopiować tą samą treść do wielu miejsc docelowych w tym samym cyklu zegara
        - tj. można napisać C7, C9, C1 gdy chcemy skopiować IP (licznik rozkazów) do wejścia X w ALU oraz wyprowadzić go na magistralę adresową wpisując do MBR



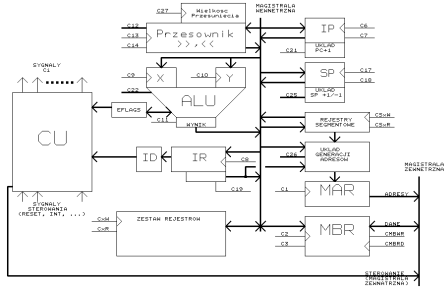
Hipotetyczny CPU

- Operacje - przyjęta notacja
  - przesłanie z rejestru A do wejścia X w ALU
    - CAR, C9
  - przesłanie z rejestru B do wejścia Y i wykonanie operacji dodawania przez ALU
    - CBR, C10, C22(+) - brak konfliktu
      - 1)na zboczu narastającym zegara działa wyprowadzenie danych przez sygnał CBR
      - 2)na zboczu opadającym następuje zapis do Y i wysterowanie ALU do operacji dodawania (ALU jako takie do swoje pracy nie wymaga sygnału zegarowego)
  - zapisanie wyniku z ALU do rejestru A
    - C11, CAW
      - uwaga! po tej operacji ustawiane są flagi - układ sterujący może to uwzględnić w następnej operacji (gdy wykonujemy ciąg: add a,b; adc c,d)
  - powyższy ciąg operacji jest realizacją fazy wykonania instrukcji ADD A,B

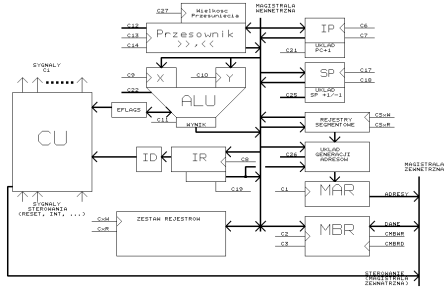


- **Hipotetyczny CPU**
  - Faza pobrania - występuje przed każdą z instrukcji
    - C7, C26, C1 wystawiamy licznik rozkazów IP na mag. adresową
      - pomijamy CScsR jako nie istotny
    - CMBWR pamięć zwróciła zawartość OPCODE do wykonania
    - C3, C8, C21
  - Faza wykonania instrukcji (pomijamy fazę pobrania - w CU jest ona zapisana tylko raz dla wszystkich instrukcji, pomijamy także fazę dekodowania OPCODE)
    - JMP PC+arg.wew
      - C19, C9 kopiujemy arg. wew zapisany w OPCODE (inaczej imm) do X
      - C7, C10 kopiujemy PC do Y i wykonujemy obliczanie PC+arg.wew
      - C11, C6 zapisujemy nową zawartość PC
    - po fazie wykonania następuje przejście do fazy sprawdzania przerwań a po niej do fazy pobrania - gdzie nowe PC ustali co ma być dalej wykonane!

Systemy Komputerowe



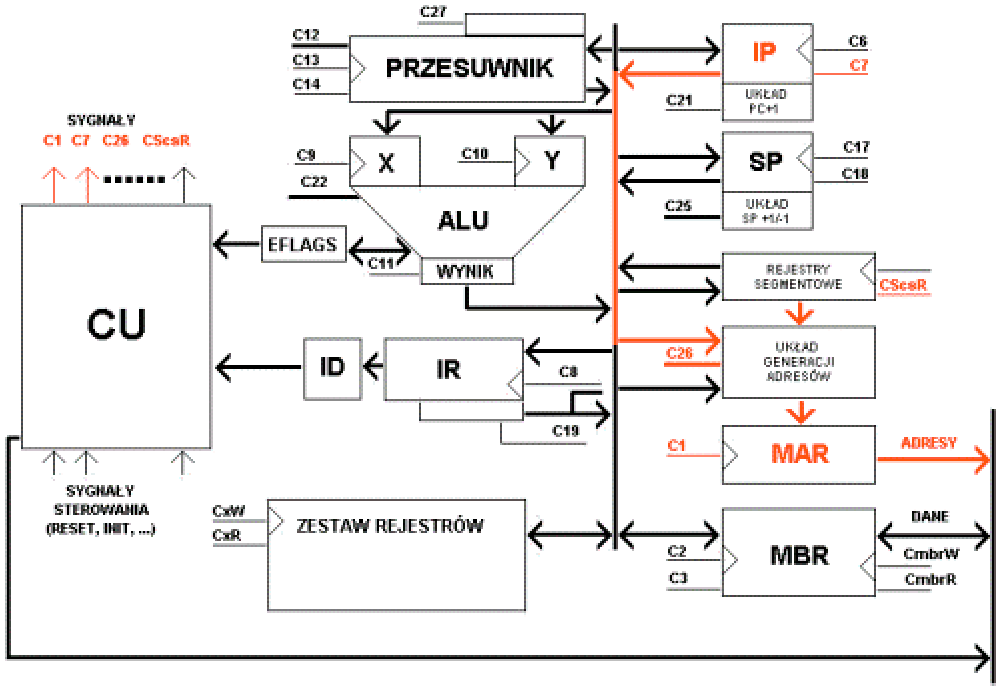
- **Hipotetyczny CPU**
  - Przesuwnik - jego rola
    - pomaga wykonać operacje których ALU nie wykonuje
      - argument do przesunięcia bitowego ładuje się przez C13
      - argument o ile bitów wykonać przesunięcie - C27
      - typ operacji ustala C12 (<<, >>, L<<, L>>, ...)
      - wynik wyprowadza na magistralę wewnętrzną - C14
  - Interaktywna pomoc
    - http://cygnus.tele.pw.edu.pl/olek/doc/syko/www/rozdzial2_2_2.html



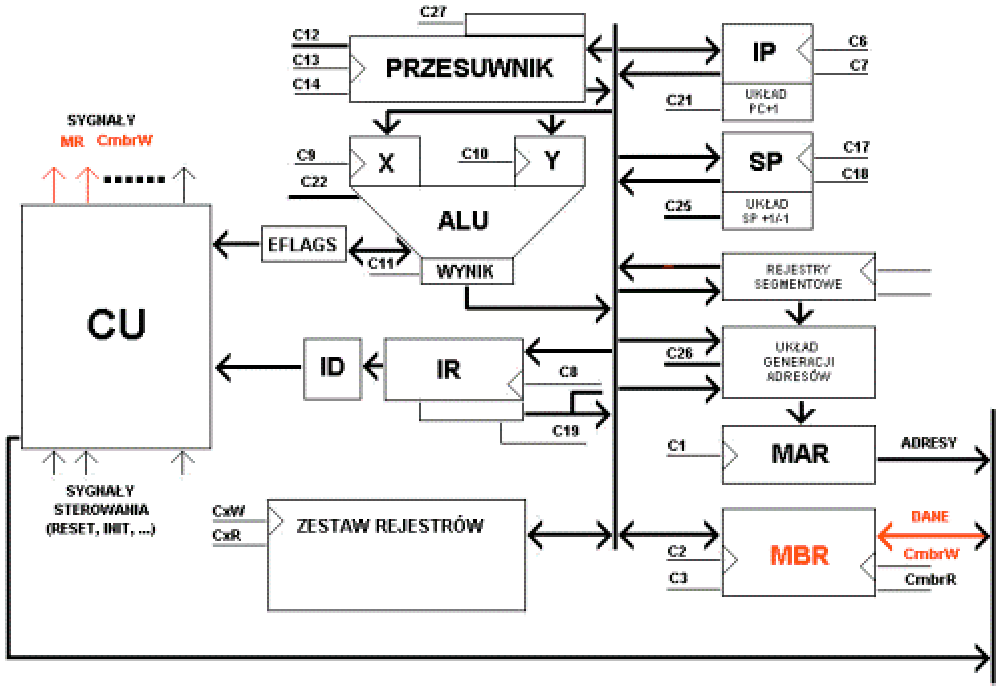
- **Hipotetyczny CPU**
  - **Działanie jednostki sterującej**
    - **wyprowadza sygnały Cxx na bazie swojego stanu**
    - **utworzona za pomocą**
      - rozbudowanej sieci połączeń bramek logicznych
        - sieć zależna od wspieranej listy instrukcji
        - w dzisiejszych czasach jeżyki typu VERILOG/VHDL upraszają proces utrzymania takiej konstrukcji np.: przy dodawaniu nowych instrukcji
      - tablicy zwanej mikro-kodem i generycznym zestawem połączonych odpowiednio bramek logicznych
        - dodanie nowej wspieranej przez CPU instrukcji nie wymaga zmiany połączeń tych bramek - zmieniamy tylko tzw. mikro-kod

Przykład realizacji przez hipotetyczny CPU wybranych instrukcji

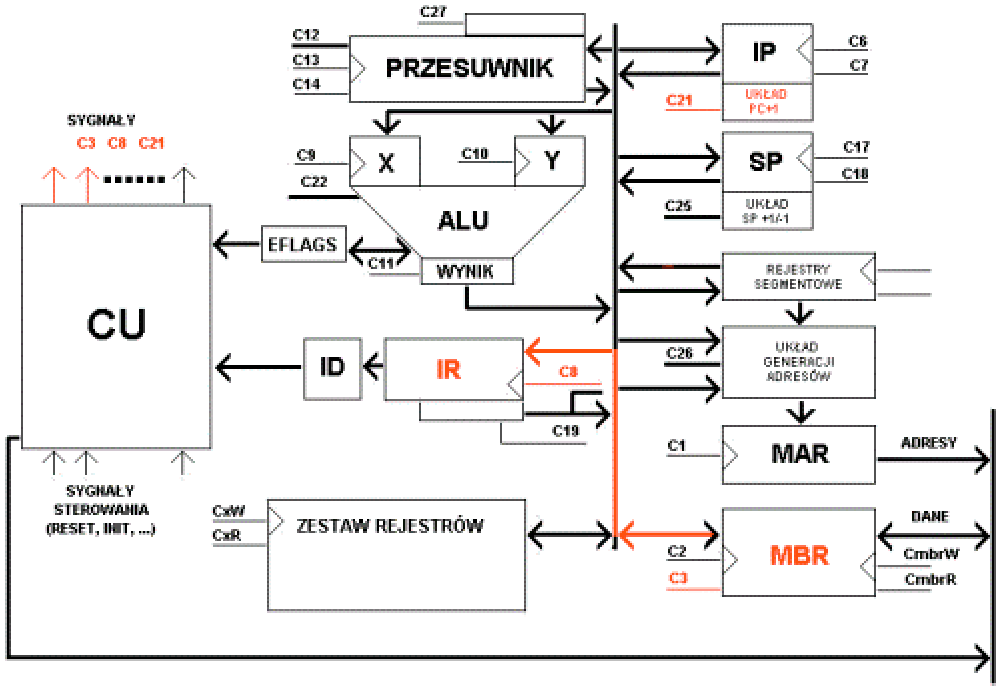
1. **FAZA POBRANIA ROZKAZU**: Wstawienie do rejestru adresowego pamięci MAR adresu kolejnego wykonywanego rozkazu.



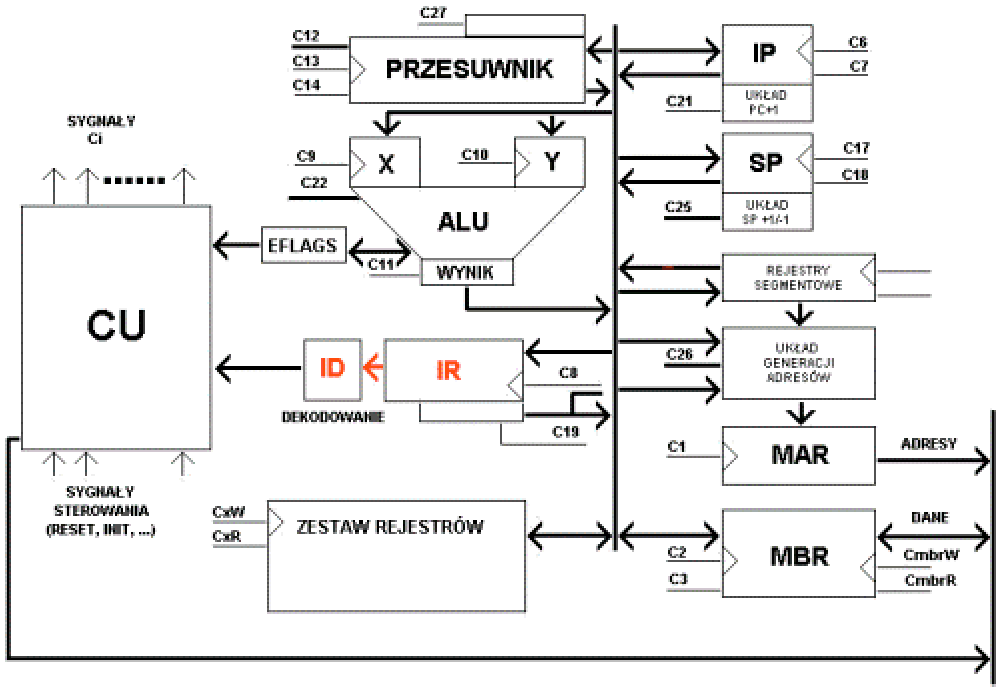
2. **FAZA POBRANIA ROZKAZU**: Odczyt kody rozkazu z pamięci. Kod rozkazu zostaje zapamiętany w rejestrze buforowym pamięci MBR.



3. **FAZA POBRANIA ROZKAZU**: Przesłanie kody rozkazu z rejestru buforowego pamięci MBR do rejestru rozkazu IR. Inkrementacja licznika programu IP.

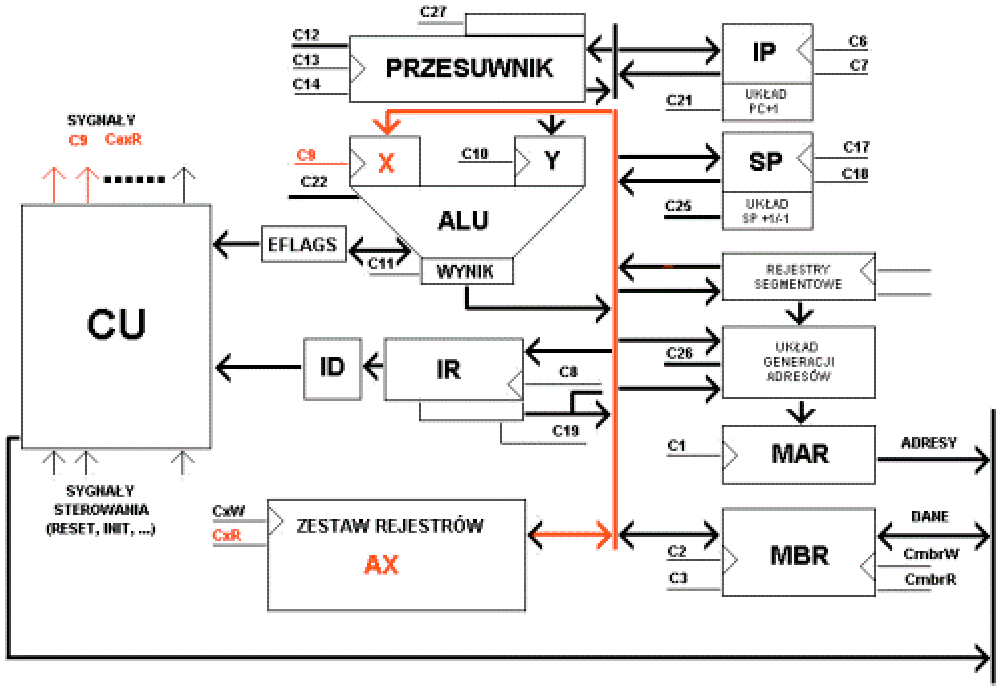


4. **FAZA POBRANIA ROZKAZU**: Dekodowanie rozkazu



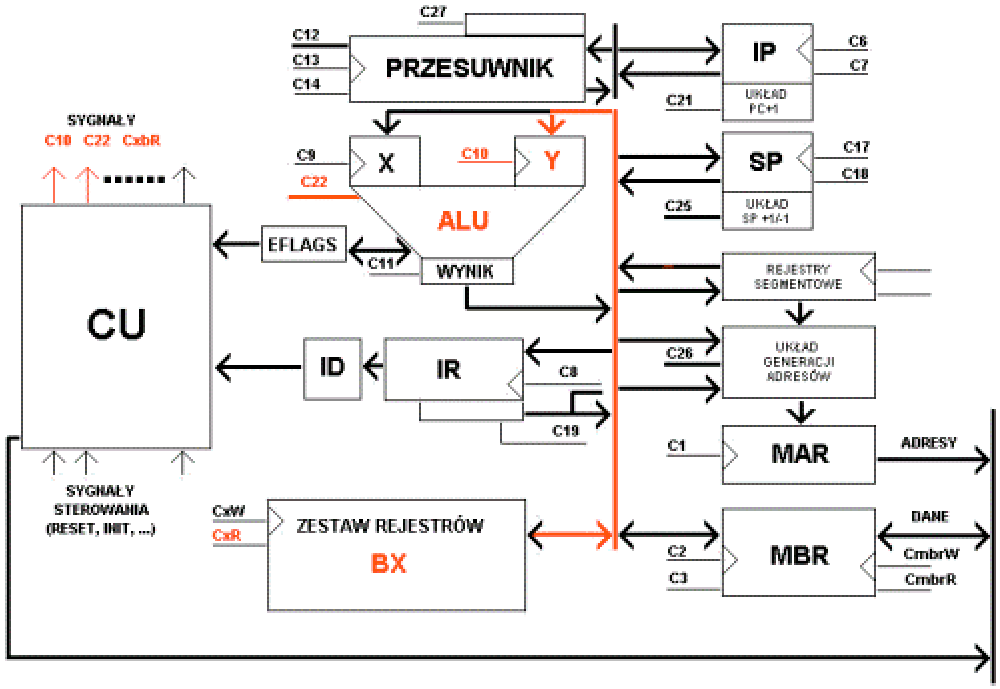
Rozkaz:
ADD AX, BX

5. FAZA WYKONYWANIA ROZKAZU: Wstawienie do rejestru X jednostki ALU zawartości rejestru AX.



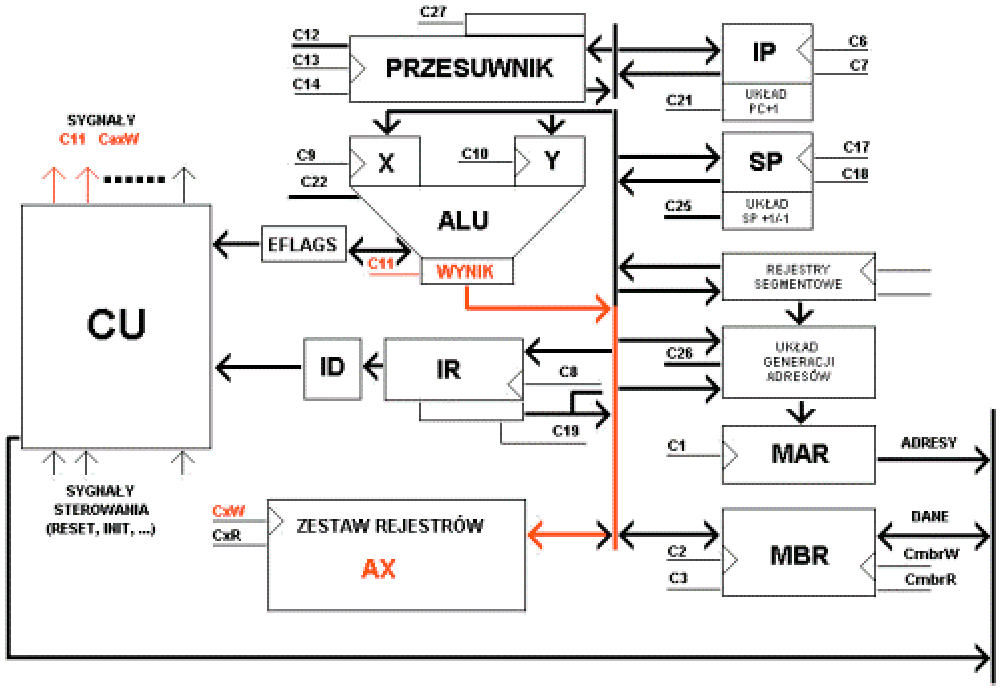
Rozkaz:
ADD AX, BX

6. FAZA WYKONYWANIA ROZKAZU: Wstawienie do rejestru Y jednostki ALU zawartości rejestru BX. Wykonanie operacji dodawania.



Rozkaz:
ADD AX, BX

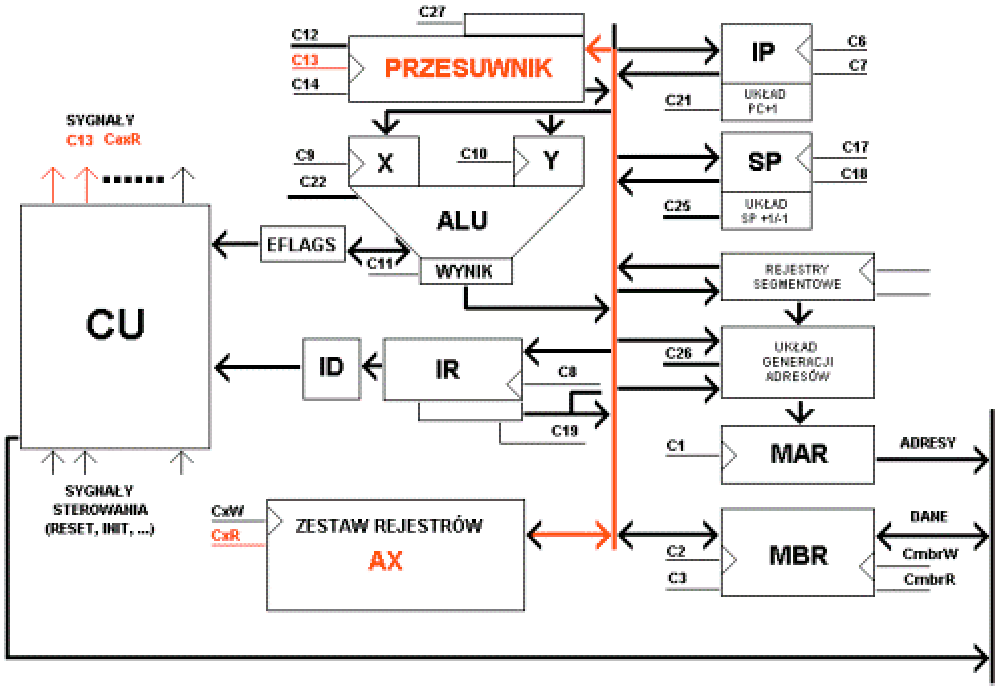
7. FAZA WYKONYWANIA ROZKAZU: Wstawienie do rejestru AX wyniku operacji dodawania, zawartej w rejestrze wynikowym jednostki ALU.



Rozkaz:
ADD AX, BX

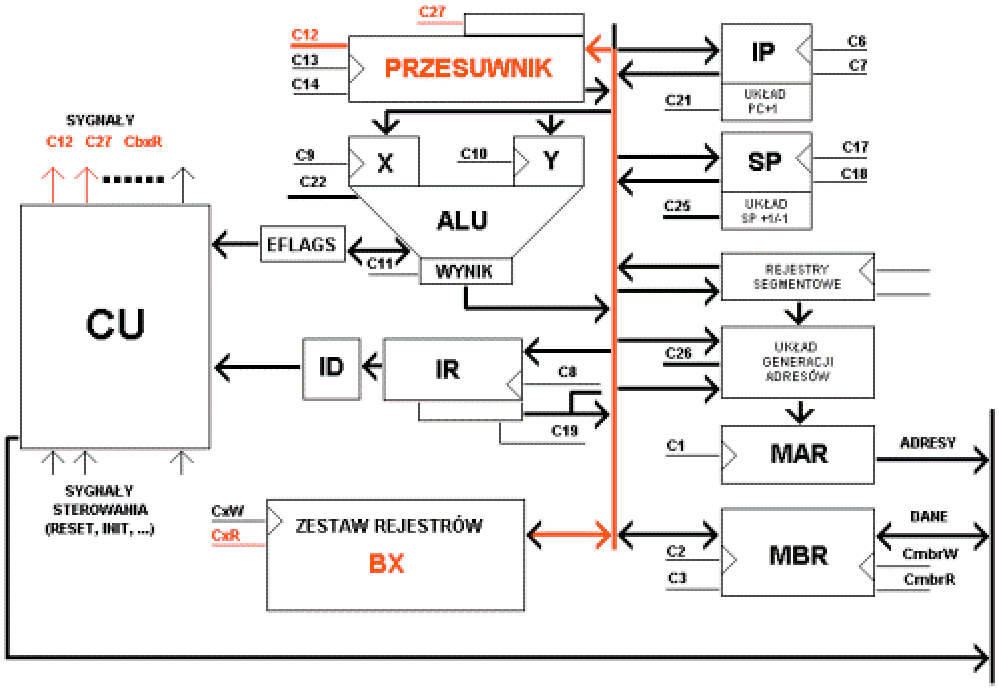
Rozkaz:
SAR AX,BX

5. FAZA WYKONYWANIA ROZKAZU: Wysłanie do układu przesuwnika zawartości rejestru AX.



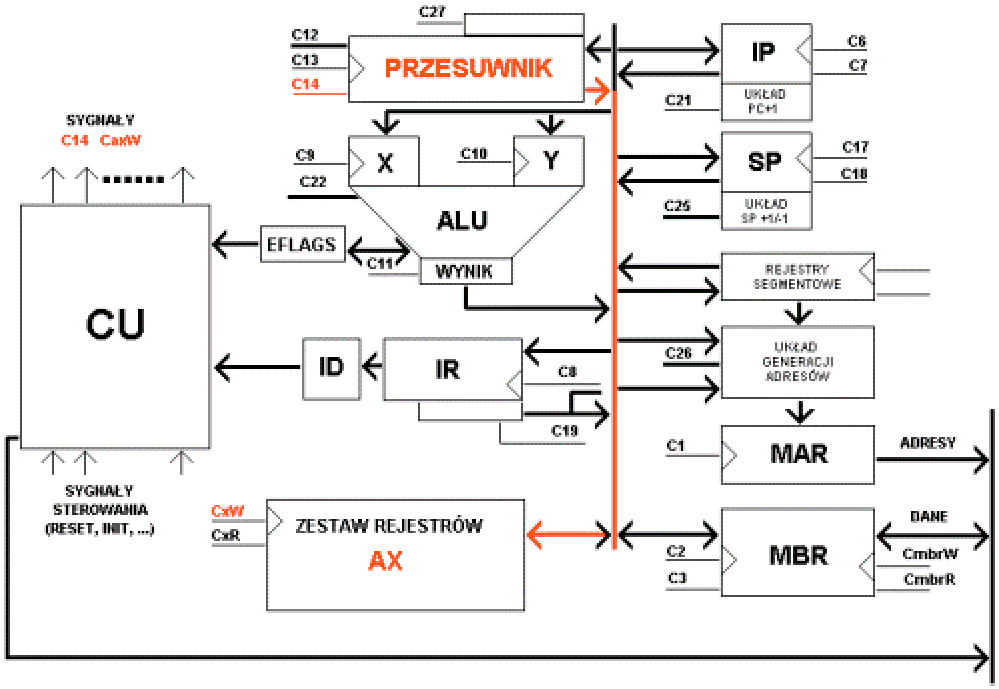
Rozkaz:
SAR AX,BX

6. FAZA WYKONYWANIA ROZKAZU: Wysłanie do układu przesuwnika zawartości rejestru BX. Przesunięcie w prawo liczby zawartej w rejestrze AX o liczbę bitów zawartą w rejestrze BX.



Rozkaz:
SAR AX,BX

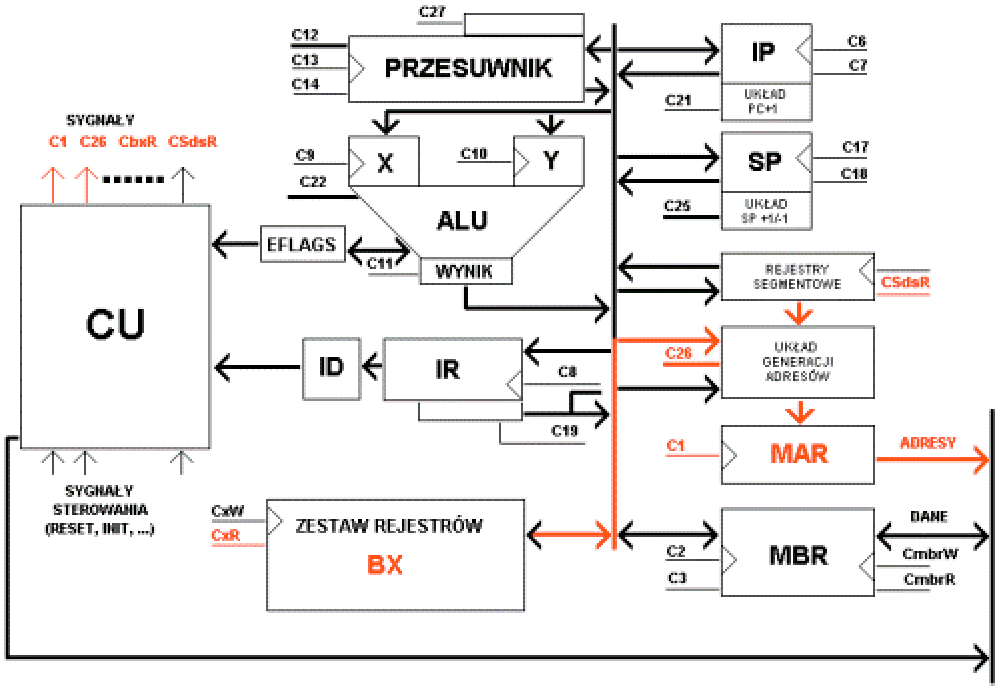
7. FAZA WYKONYWANIA ROZKAZU: Wysłanie do rejestru AX wyniku operacji przesuwania.



Rozkaz:
SAR AX,BX

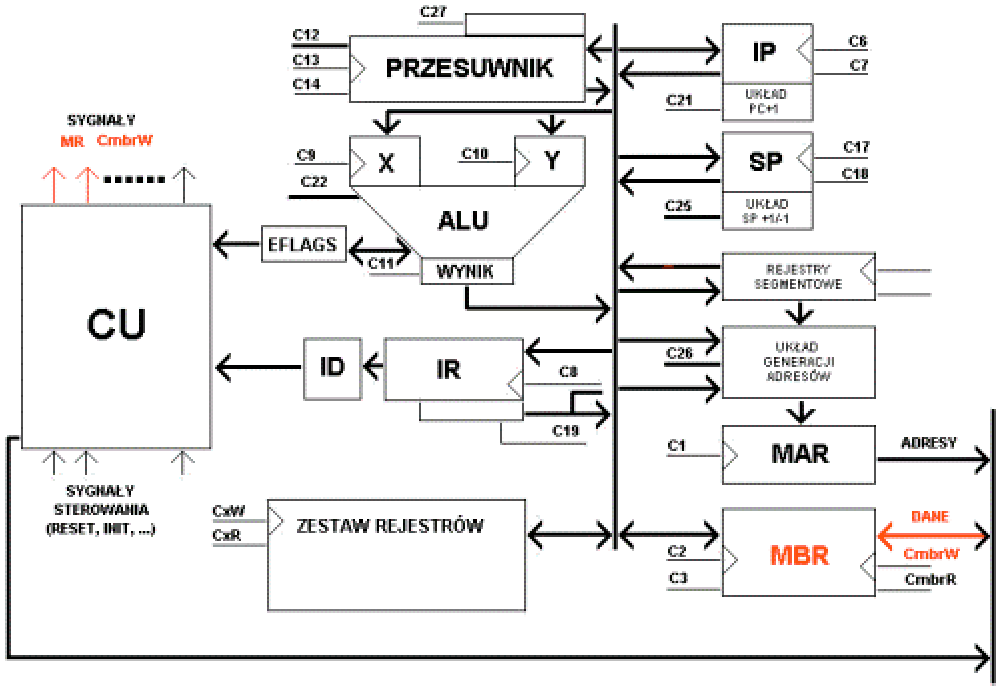
Rozkaz:
AND AX, [BX]

5. FAZA WYKONYWANIA ROZKAZU: Wstawienie do rejestru adresowego pamięci MAR adresu znajdującego się w rejestrze BX



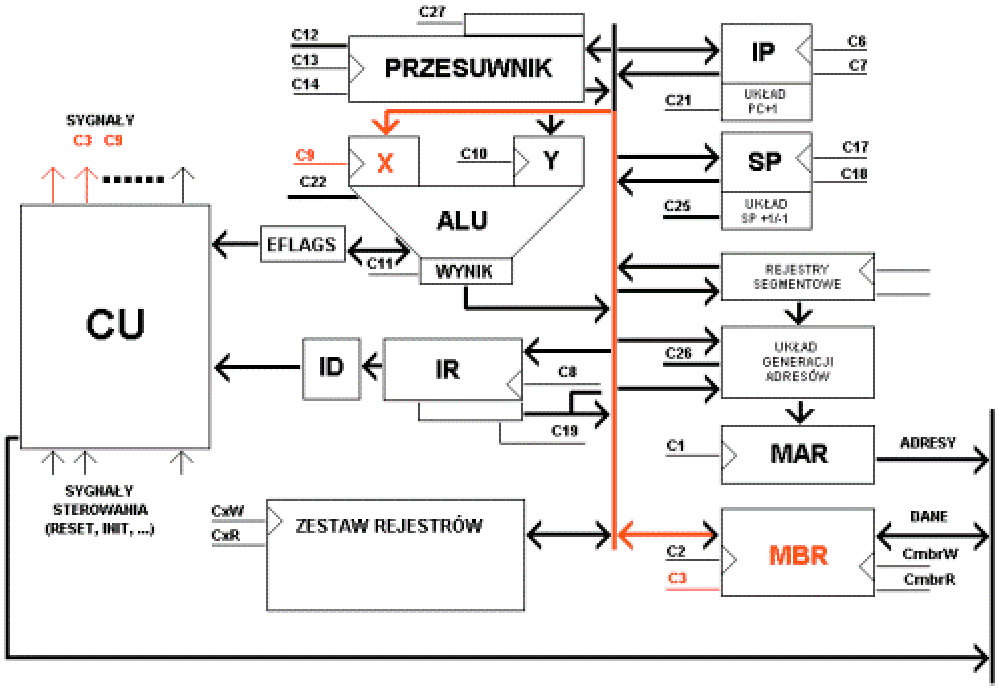
Rozkaz:
AND AX, [BX]

6. FAZA WYKONYWANIA ROZKAZU: Odczyt pamięci. Odczytana wartość (w tym przypadku jeden z argumentów rozkazu) zostaje zapisana w rejestrze buforowym pamięci MBR.



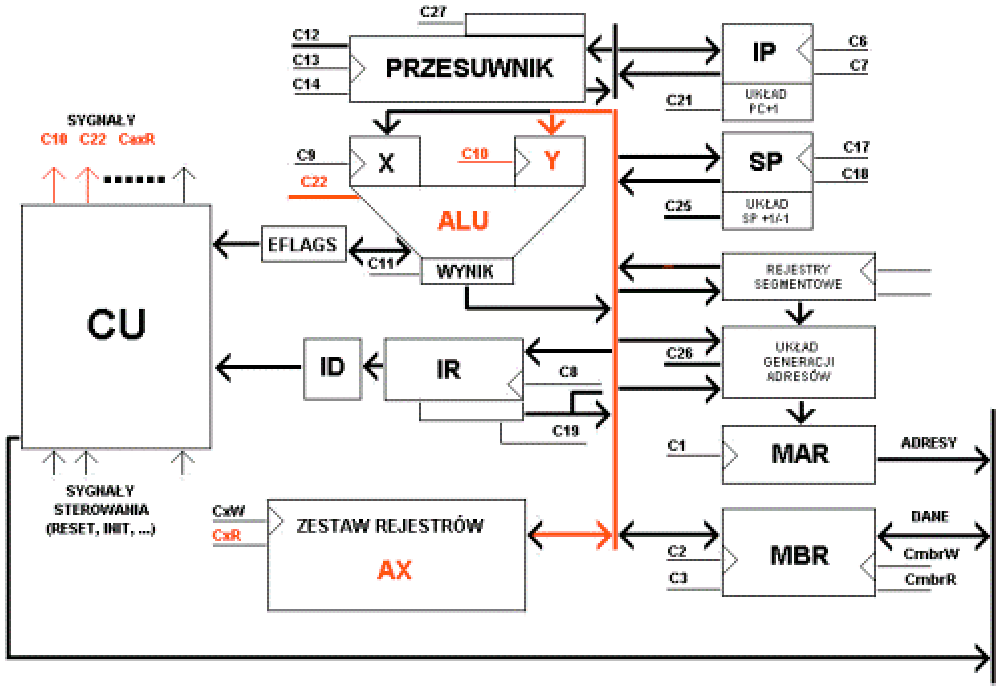
Rozkaz:
AND AX, [BX]

7. FAZA WYKONYWANIA ROZKAZU: Przesłanie argumentu rozkazu zawartego w rejestrze buforowym pamięci MBR do rejestru X układu ALU.



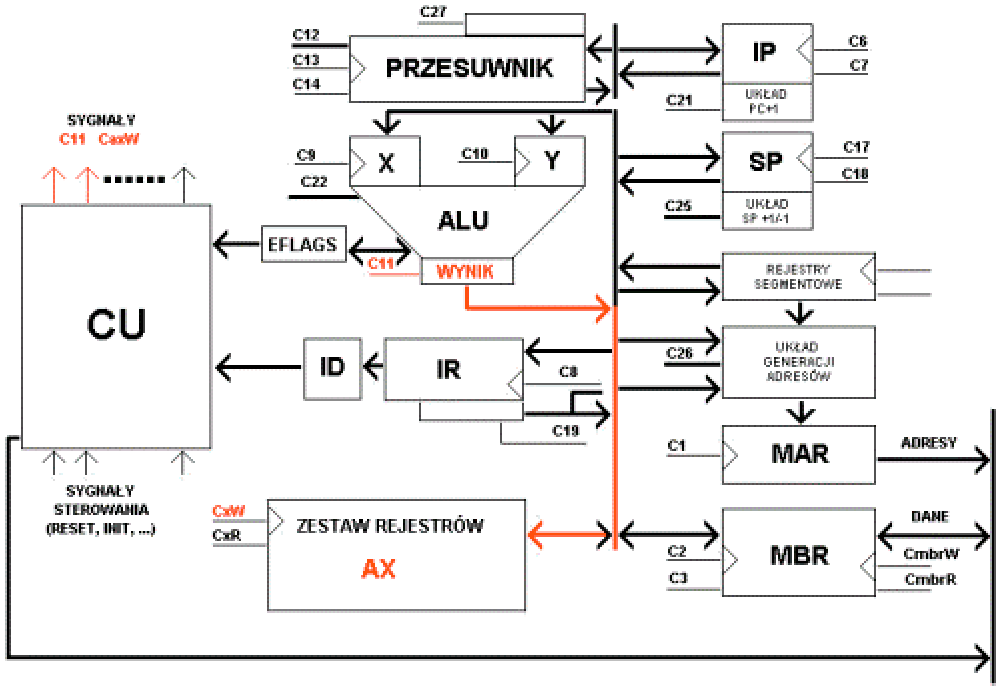
Rozkaz:
AND AX, [BX]

8. FAZA WYKONYWANIA ROZKAZU: Przesłanie drugiego argumentu rozkazu z rejestru AX do rejestru Y jednostki ALU. Wykonanie operacji iloczynu logicznego.



Rozkaz:
AND AX, [BX]

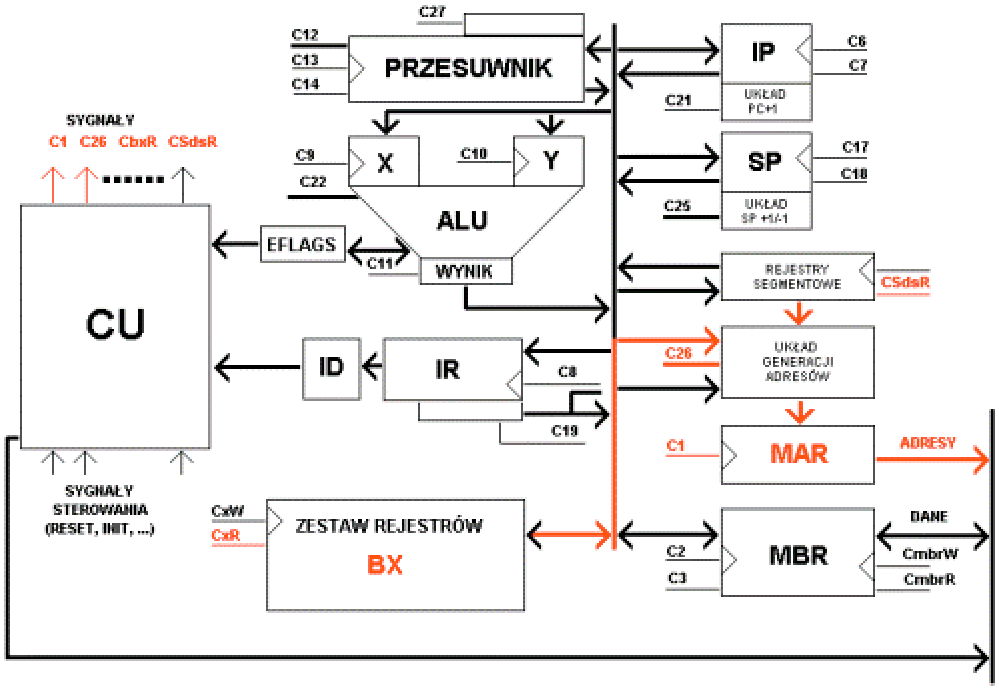
9. FAZA WYKONYWANIA ROZKAZU: Przesłanie wyniku operacji iloczynu logicznego z rejestru wynikowego jednostki ALU do rejestru AX



Rozkaz:
AND AX, [BX]

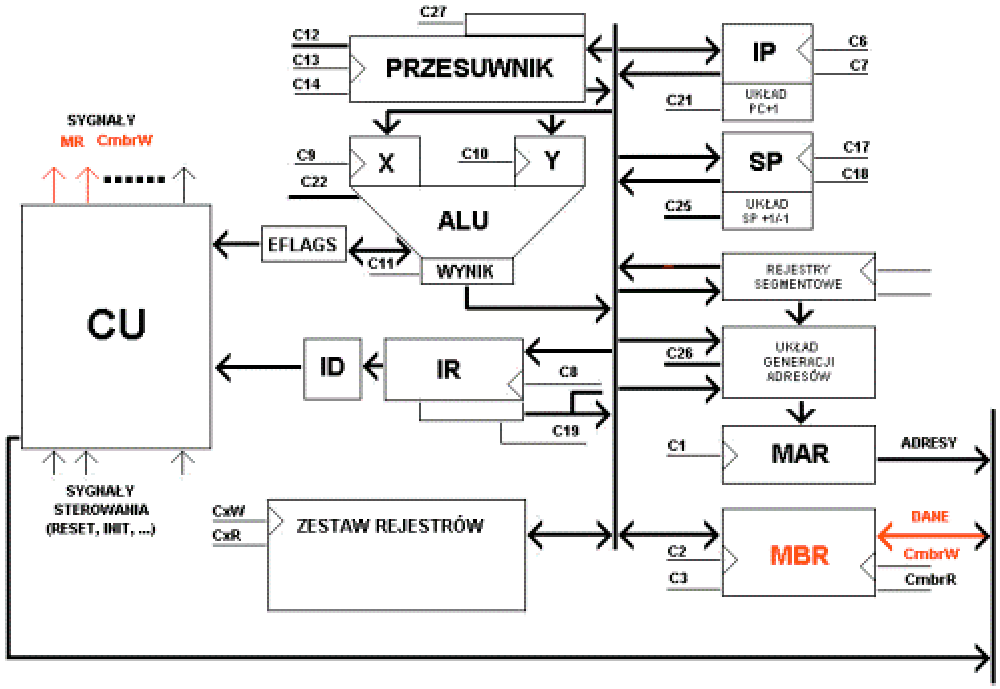
Rozkaz:
ADD [BX], AX

5. FAZA WYKONYWANIA ROZKAZU: Wstawienie do rejestru adresowego pamięci MAR adresu znajdującego się w rejestrze BX



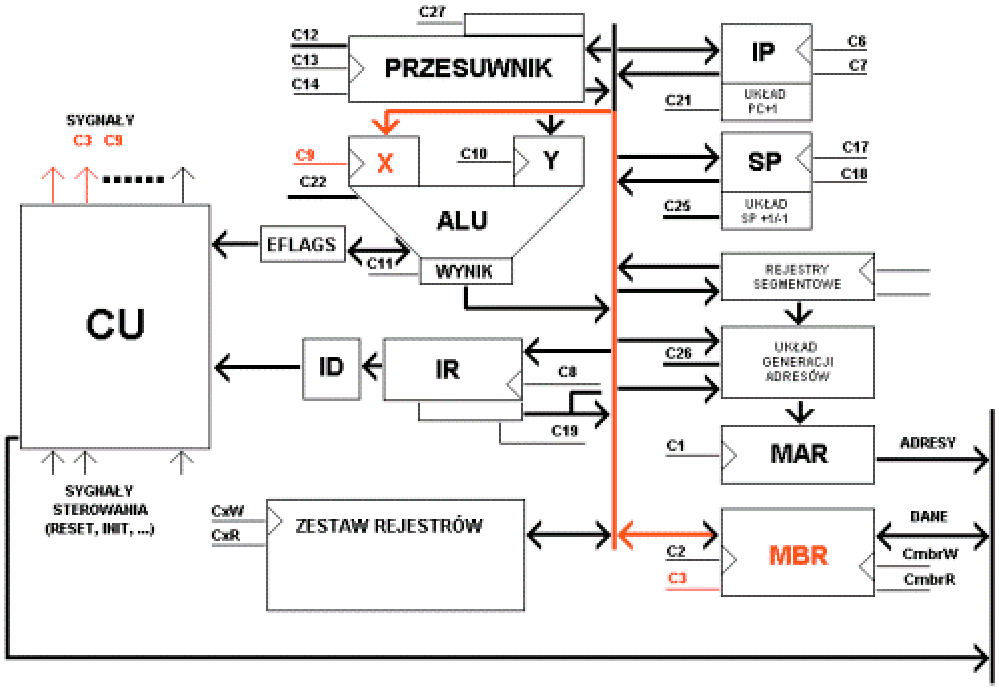
Rozkaz:
ADD [BX], AX

6. FAZA WYKONYWANIA ROZKAZU: Odczyt pamięci. Odczytana wartość (w tym przypadku jeden z argumentów rozkazu) zostaje zapisana w rejestrze buforowym pamięci MBR.



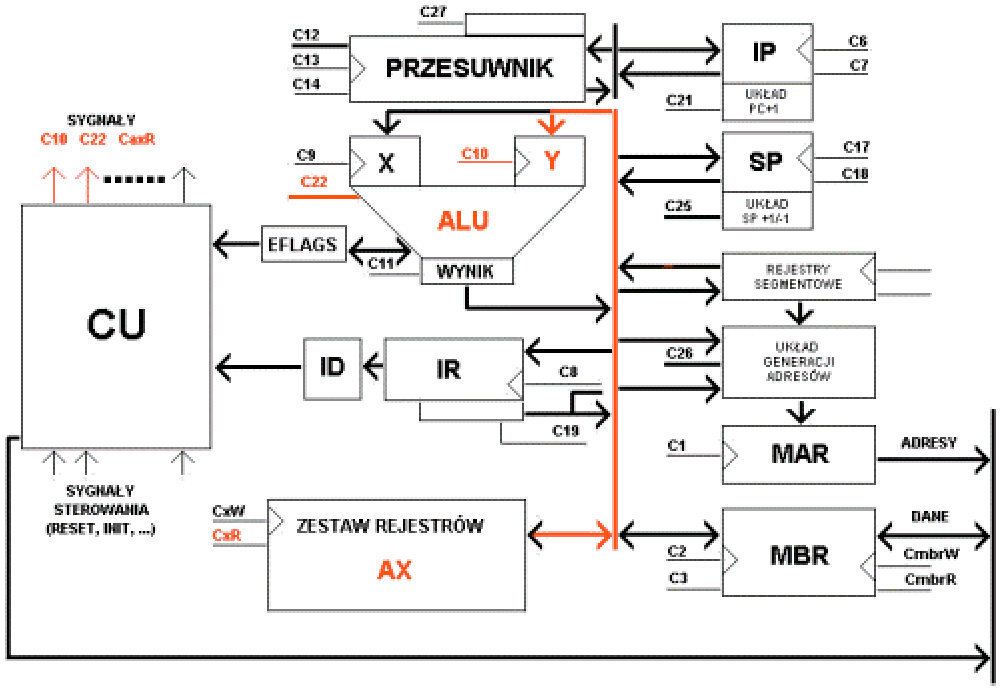
Rozkaz:
ADD [BX], AX

7. FAZA WYKONYWANIA ROZKAZU: Przesłanie argumentu rozkazu zawartego w rejestrze buforowym pamięci MBR do rejestru X układu ALU.



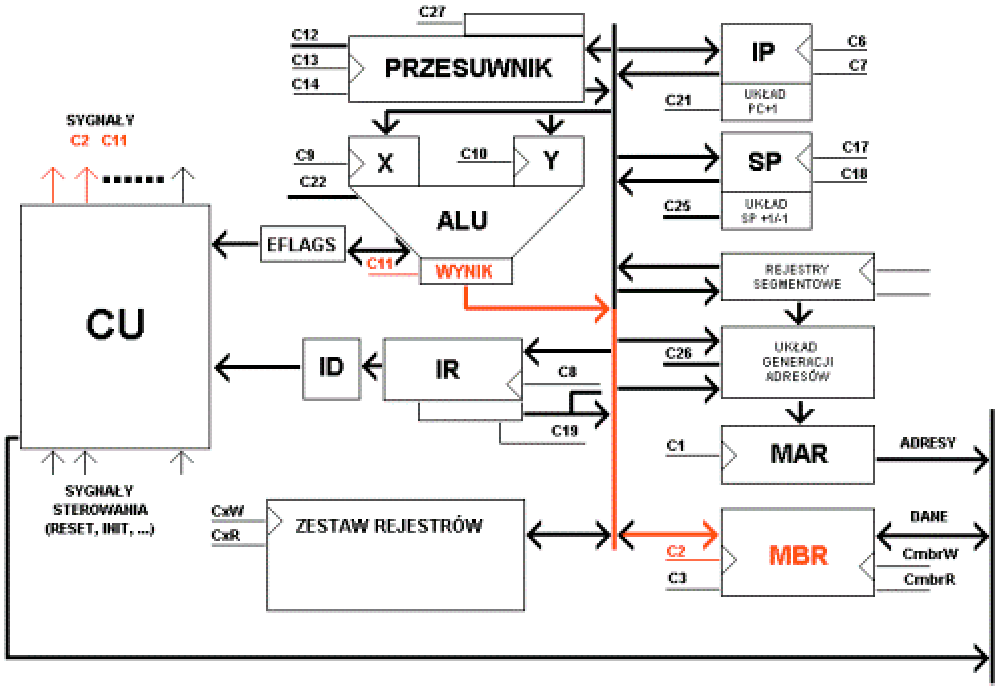
Rozkaz:
ADD [BX], AX

8. FAZA WYKONYWANIA ROZKAZU: Przesłanie drugiego argumentu rozkazu z rejestru AX do rejestru Y jednostki ALU. Wykonanie operacji dodawania.



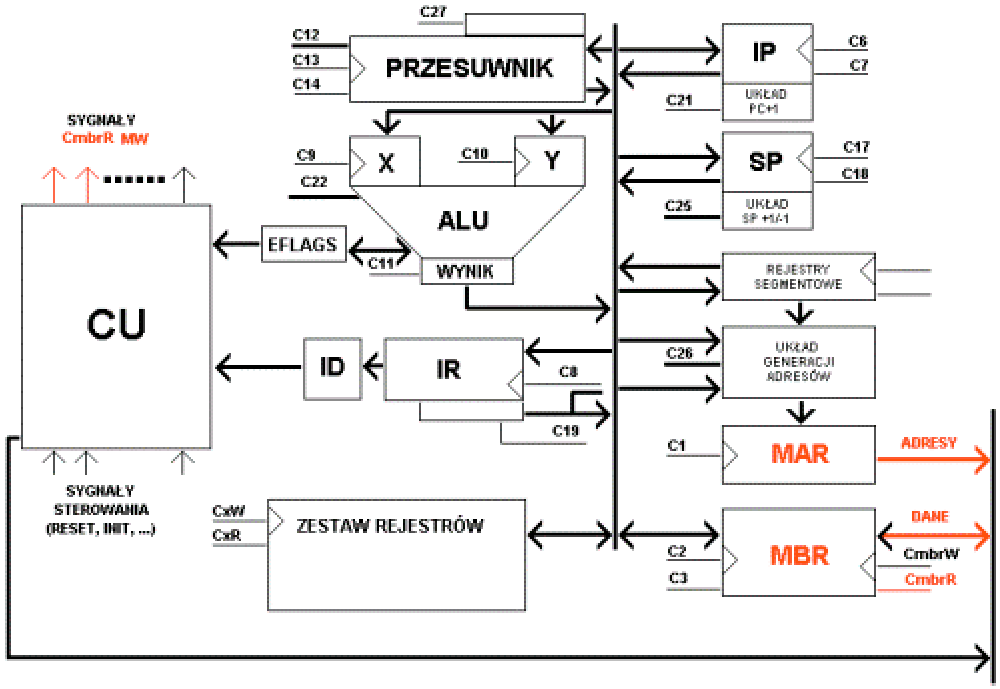
Rozkaz:
ADD [BX], AX

9. FAZA WYKONYWANIA ROZKAZU: Przesłanie wyniku operacji dodawania z rejestru wynikowego jednostki ALU do rejestru buforowego pamięci MBR



Rozkaz:
ADD [BX], AX

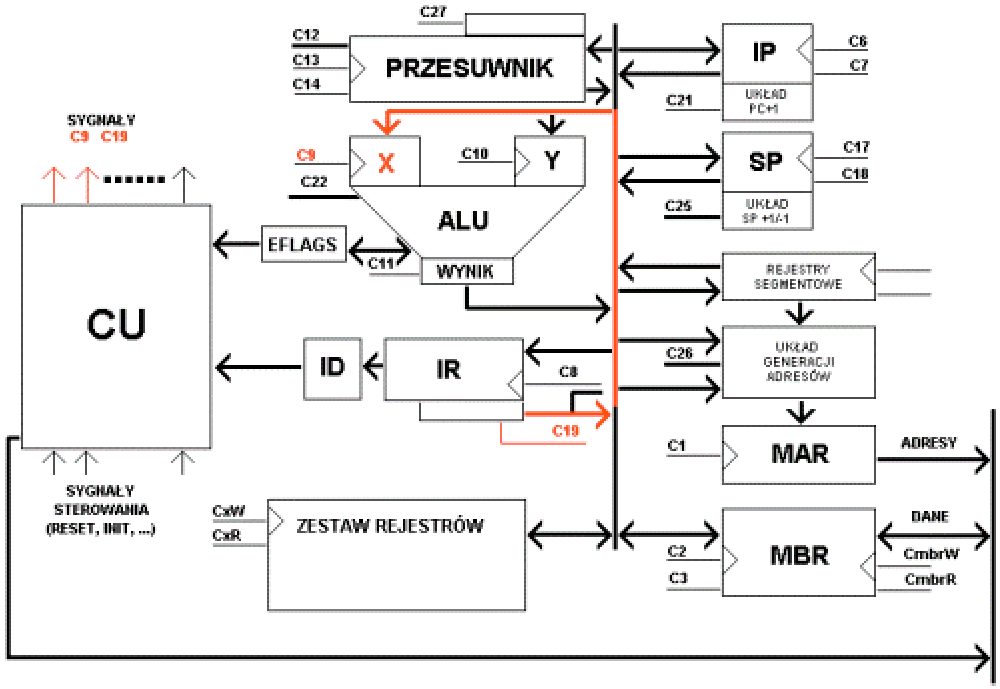
10. FAZA WYKONYWANIA ROZKAZU: Zapis do pamięci. Zawartość rejestru buforowego pamięci MBR zostaje zapisana pod adresem zawartym w rejestrze adresowym MAR



Rozkaz:
ADD [BX], AX

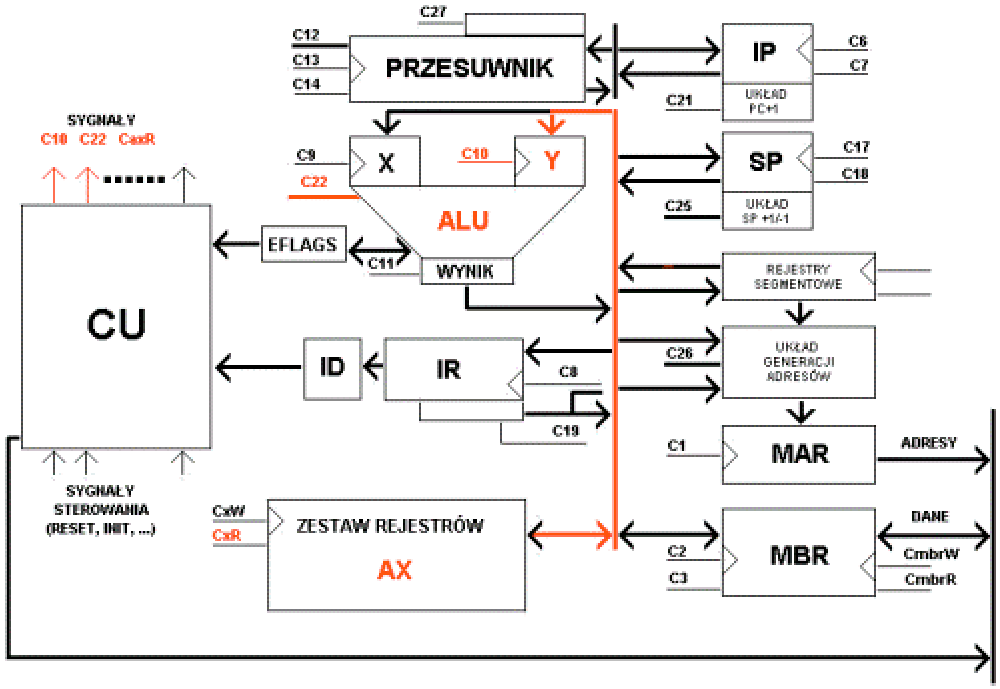
Rozkaz:
ADD AX, arg. wbudowany

5. FAZA WYKONYWANIA ROZKAZU: Wstawienie do rejestru X jednostki ALU zawartości pola natychmiastowego rejestru rozkazu IR.



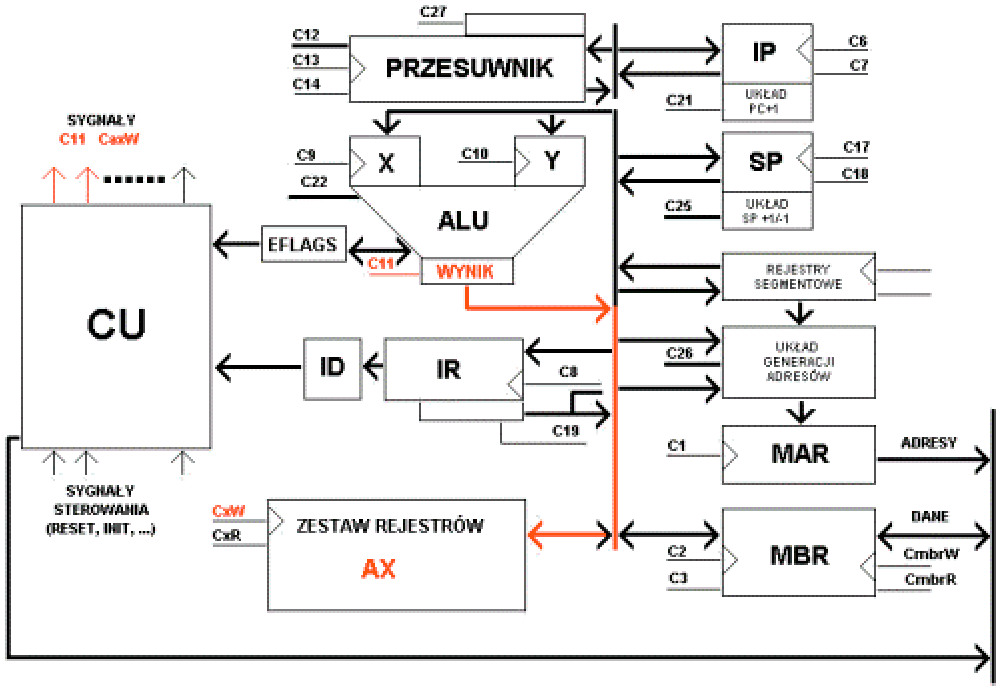
Rozkaz:
ADD AX, arg. wbudowany

6. FAZA WYKONYWANIA ROZKAZU: Wstawienie do rejestru Y jednostki ALU zawartości rejestru AX. Wykonanie operacji dodawania.



Rozkaz:
ADD AX, arg. wbudowany

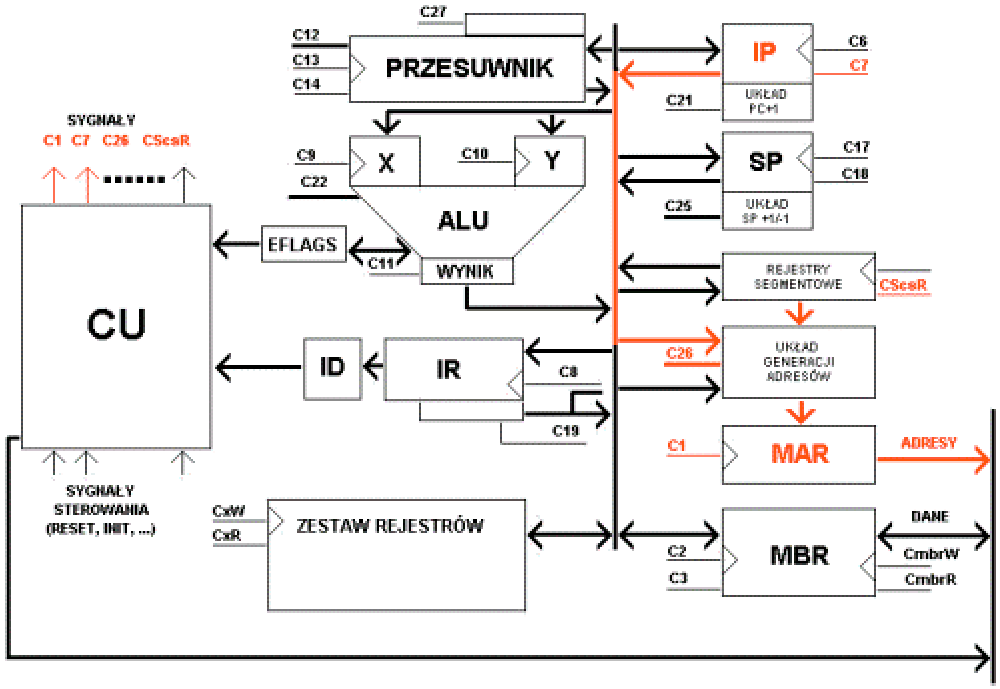
7. FAZA WYKONYWANIA ROZKAZU: Wstawienie do rejestru AX wyniku operacji dodawania, zawartej w rejestrze wynikowym jednostki ALU.



Rozkaz:
ADD AX, arg. wbudowany

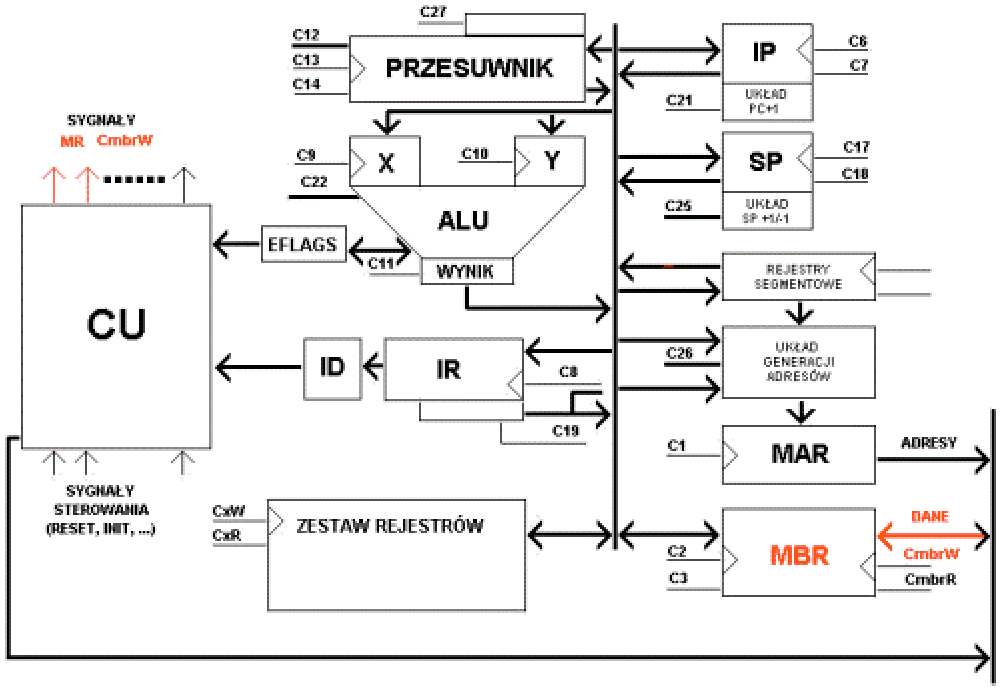
Rozkaz:
ADD AX, arg. zewnętrzny

5. FAZA WYKONYWANIA ROZKAZU: Wstawienie do rejestru adresowego pamięci MAR adresu znajdującego się w pamięci programu bezpośrednio za kodem rozkazu.



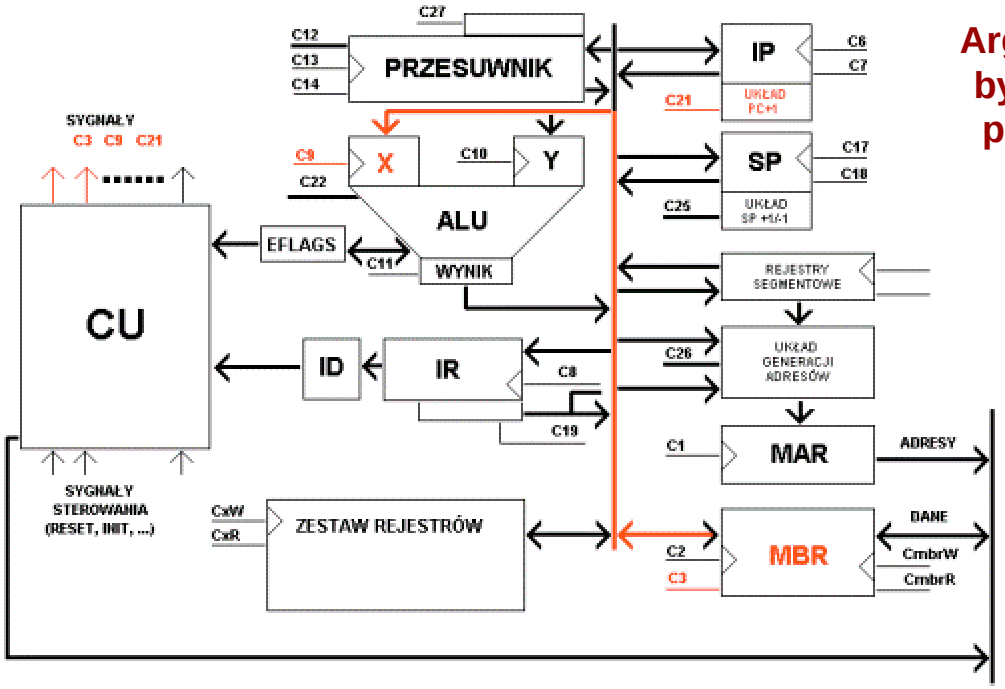
Rozkaz:
ADD AX, arg. zewnętrzny

6. FAZA WYKONYWANIA ROZKAZU: Odczyt pamięci. Odczytana wartość (w tym przypadku jeden z argumentów rozkazu) zostaje zapisana w rejestrze buforowym pamięci MBR.



Rozkaz:
ADD AX, arg. zewnętrzny

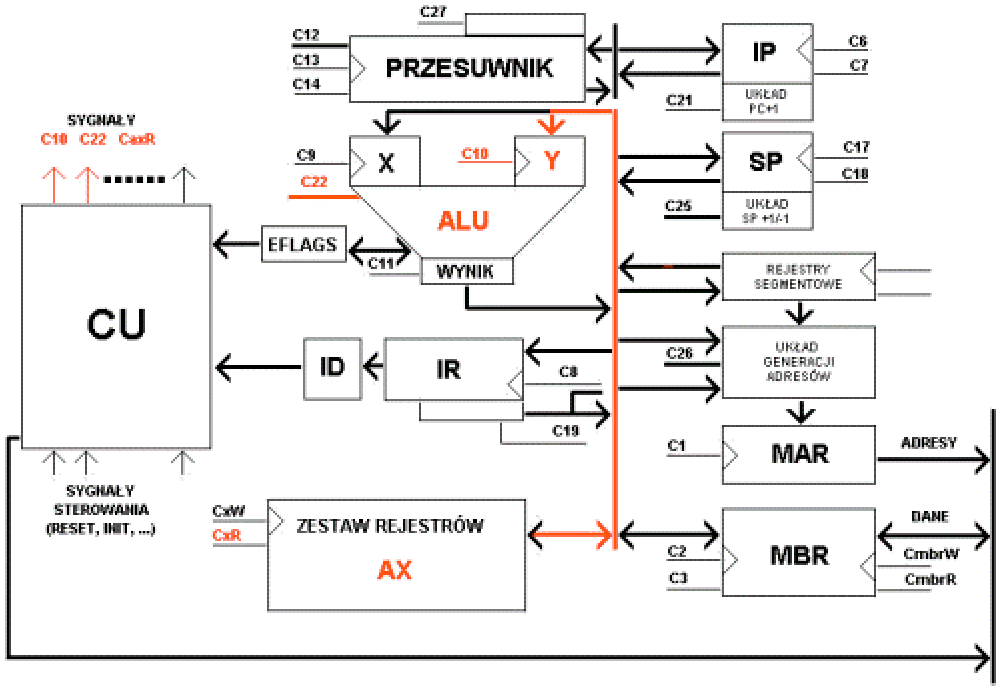
7. FAZA WYKONYWANIA ROZKAZU: Przesłanie argumentu rozkazu zawartego w rejestrze buforowym pamięci MBR do rejestru X układu ALU.



Arg.zewnętrzny -> musi być uaktywnione C21 poza fazą pobrania!!!

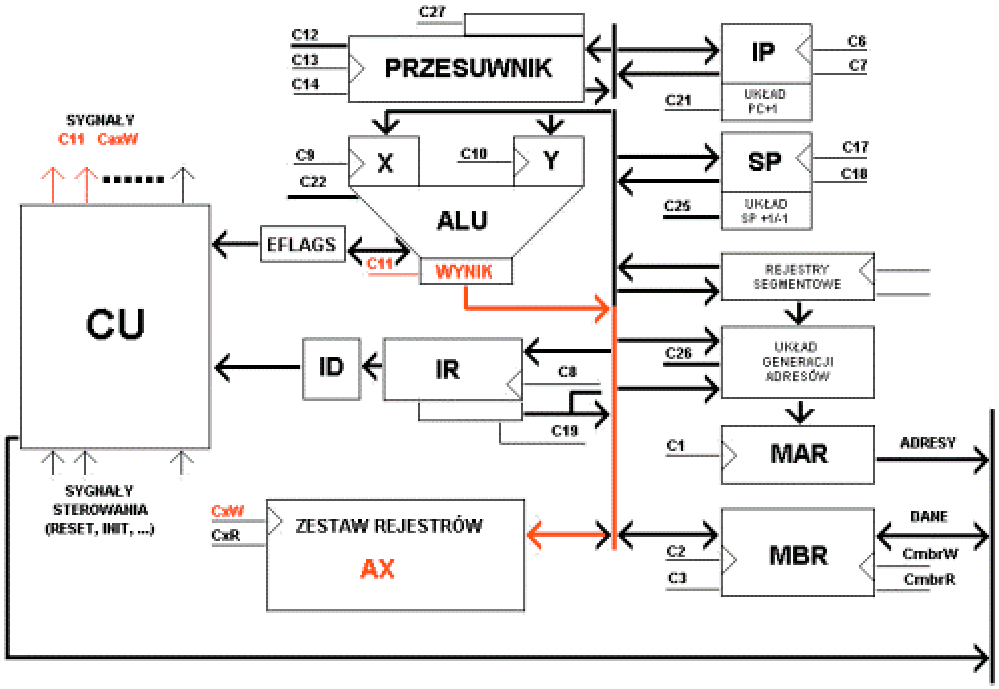
Rozkaz:
ADD AX, arg. zewnętrzny

8. FAZA WYKONYWANIA ROZKAZU: Przesłanie drugiego argumentu rozkazu z rejestru AX do rejestru Y jednostki ALU. Wykonanie operacji dodawania.



Rozkaz:
ADD AX, arg. zewnętrzny

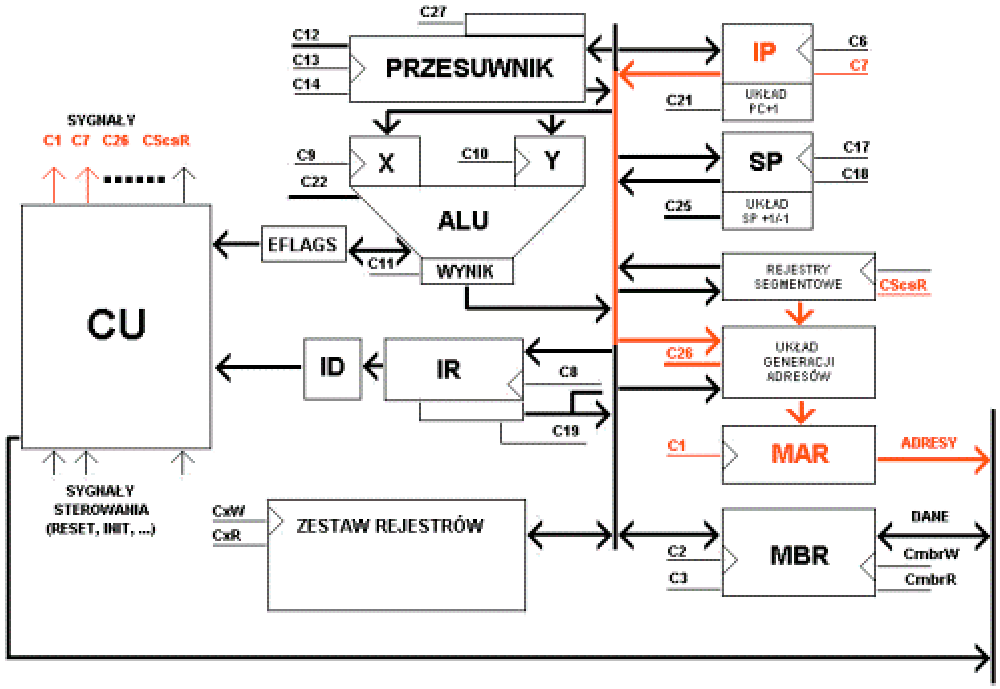
9. FAZA WYKONYWANIA ROZKAZU: Wysłanie wyniku operacji dodawania z rejestru wynikowego jednostki ALU do rejestru AX



Rozkaz:
ADD AX, arg. zewnętrzny

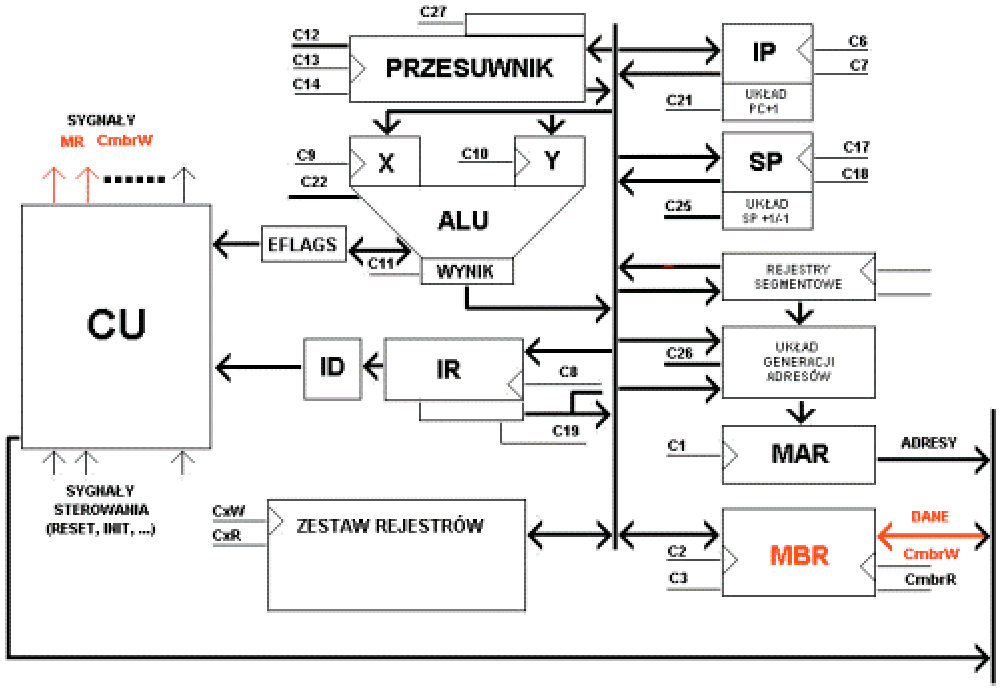
Rozkaz:
ADD [arg. zewnętrzny], AX

5. FAZA WYKONYWANIA ROZKAZU: Wstawienie do rejestru adresowego pamięci MAR adresu znajdującego się w pamięci programu bezpośrednio za kodem rozkazu.



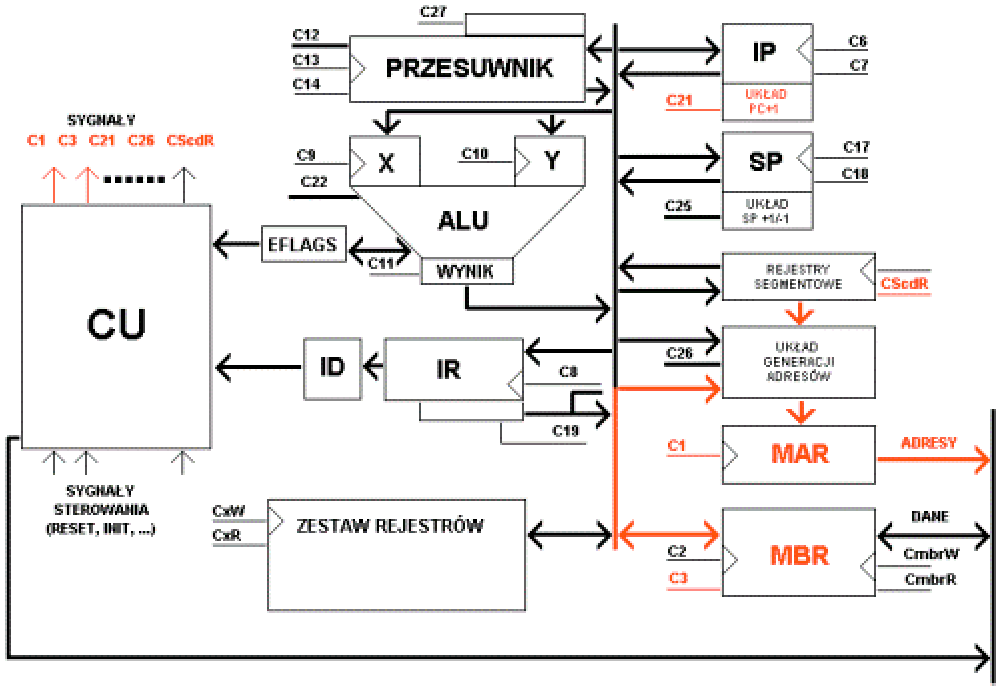
Rozkaz:
ADD [arg. zewnętrzny], AX

6. FAZA WYKONYWANIA ROZKAZU: Odczyt pamięci. Odczytana wartość (w tym przypadku adres jednego z argumentów rozkazu) zostaje zapisana w rejestrze buforowym pamięci MBR.



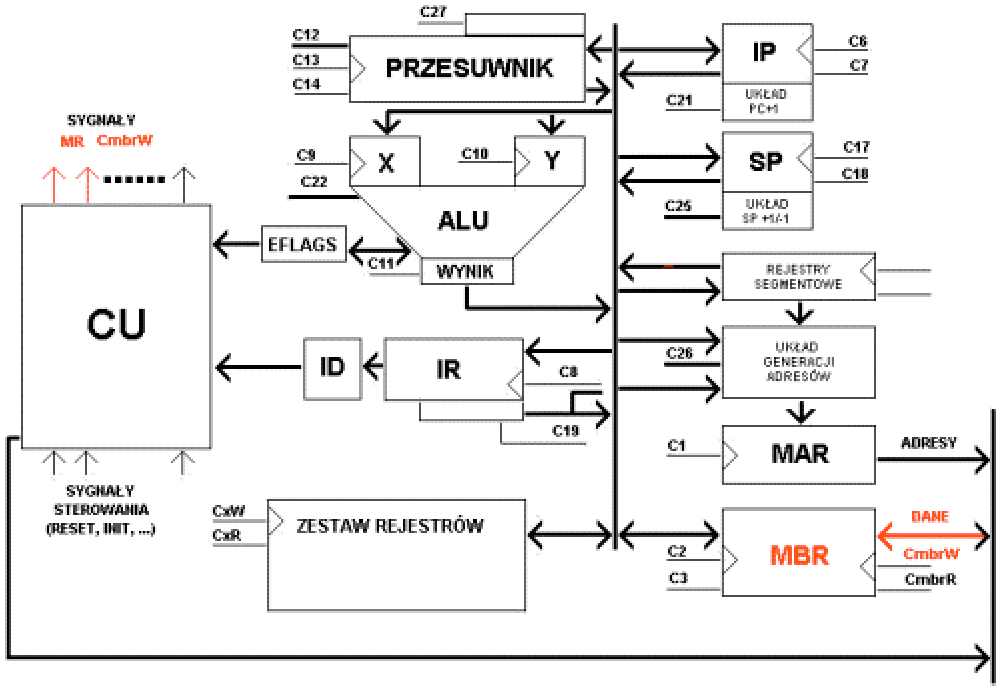
Rozkaz:
ADD [arg. zewnętrzny], AX

7. FAZA WYKONYWANIA ROZKAZU: Przesłanie zawartości rejestru buforowego pamięci MBR do rejestru adresowego pamięci MAR.



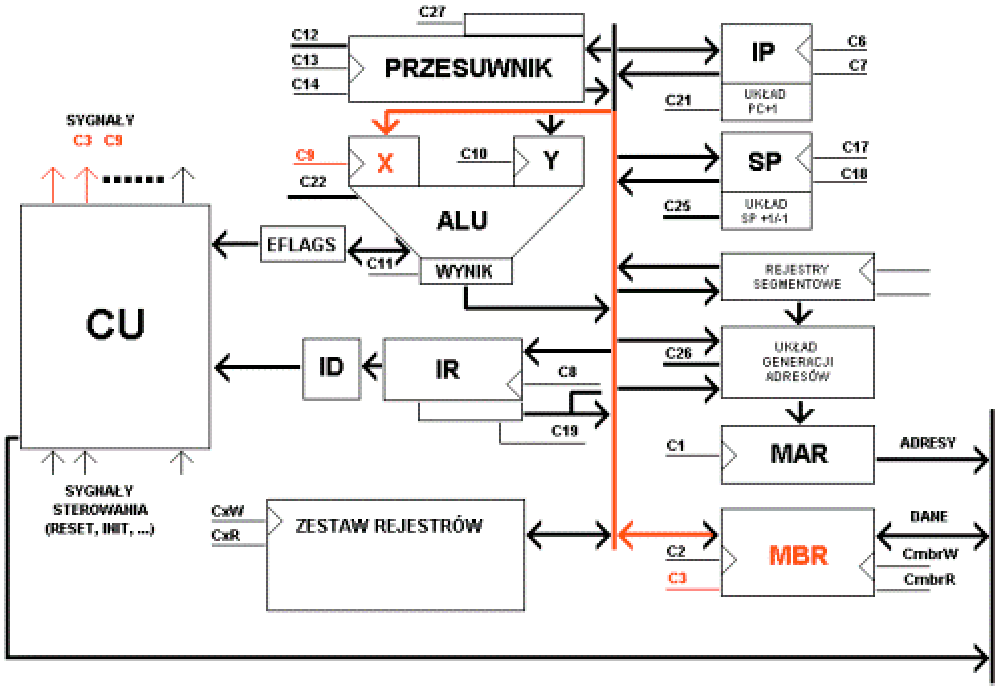
Rozkaz:
ADD [arg. zewnętrzny], AX

8. FAZA WYKONYWANIA ROZKAZU: Odczyt pamięci. Odczytana wartość (w tym przypadku jeden z argumentów rozkazu) zostaje zapisana w rejestrze buforowym pamięci MBR.



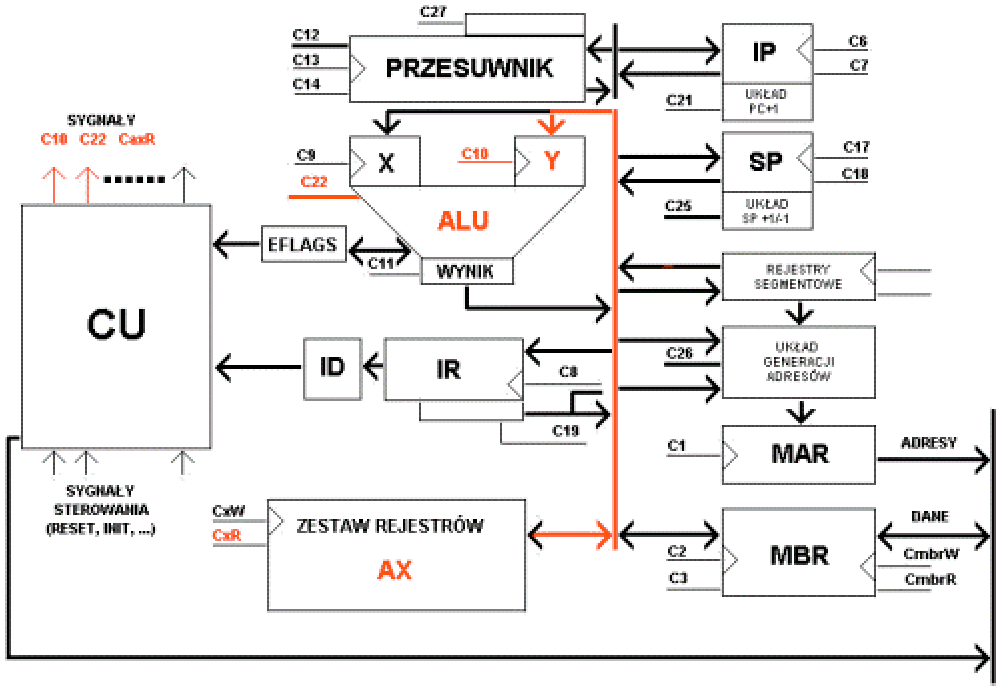
Rozkaz:
ADD [arg. zewnętrzny], AX

9. FAZA WYKONYWANIA ROZKAZU: Przesłanie argumentu rozkazu zawartego w rejestrze buforowym pamięci MBR do rejestru X układu ALU.



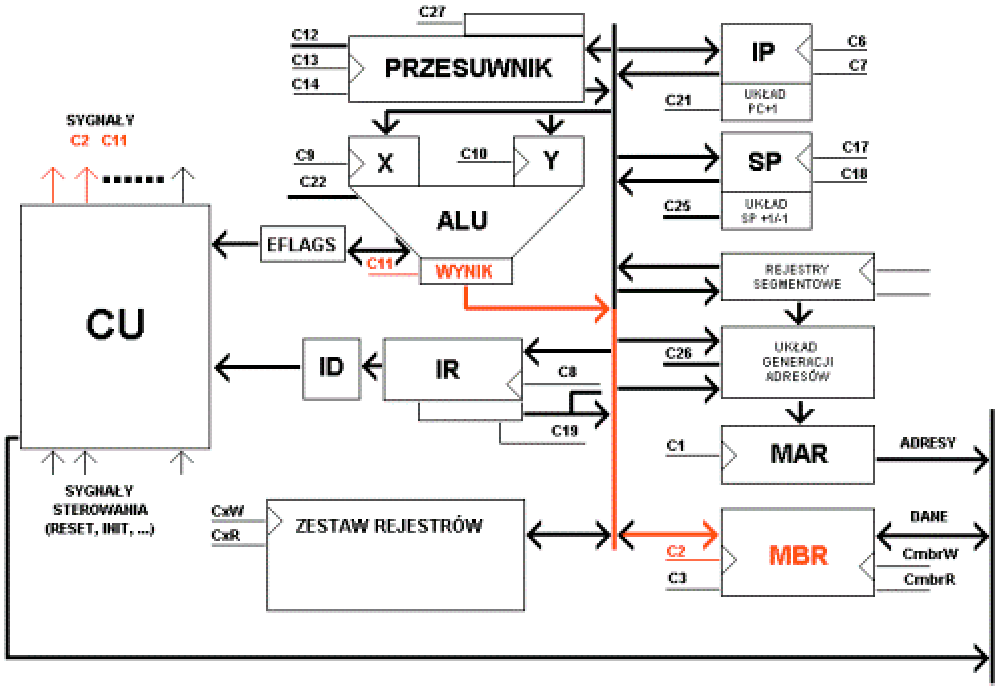
Rozkaz:
ADD [arg. zewnętrzny], AX

10. FAZA WYKONYWANIA ROZKAZU: Przesłanie drugiego argumentu rozkazu z rejestru AX do rejestru Y jednostki ALU. Wykonanie operacji dodawania.



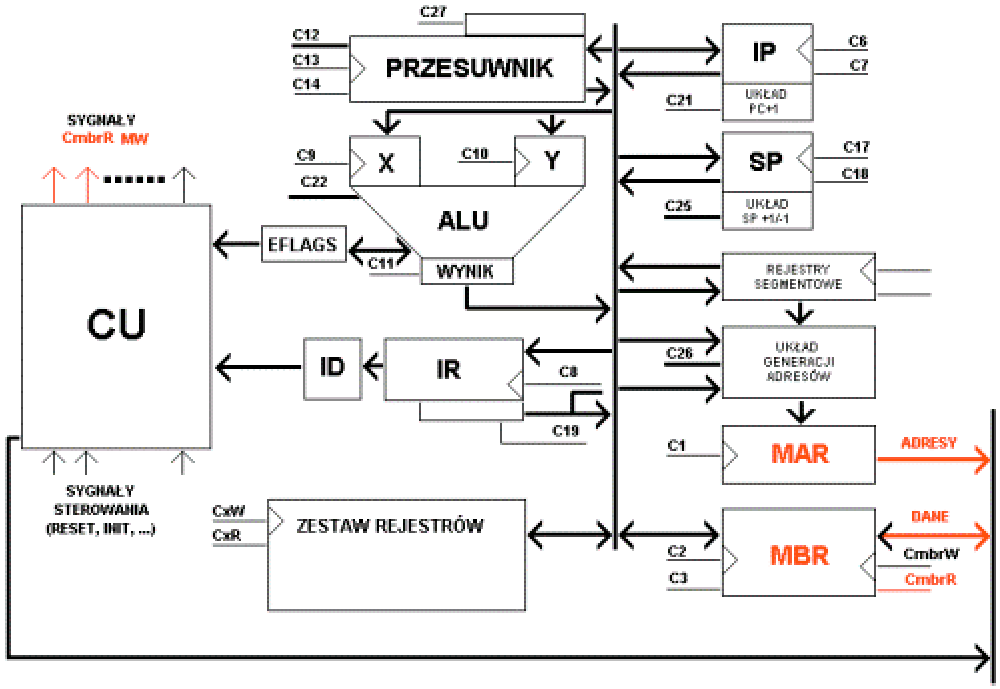
Rozkaz:
ADD [arg. zewnętrzny], AX

11. FAZA WYKONYWANIA ROZKAZU: Przesłanie wyniku operacji dodawania z rejestru wynikowego jednostki ALU do rejestru buforowego pamięci MBR



Rozkaz:
ADD [arg. zewnętrzny], AX

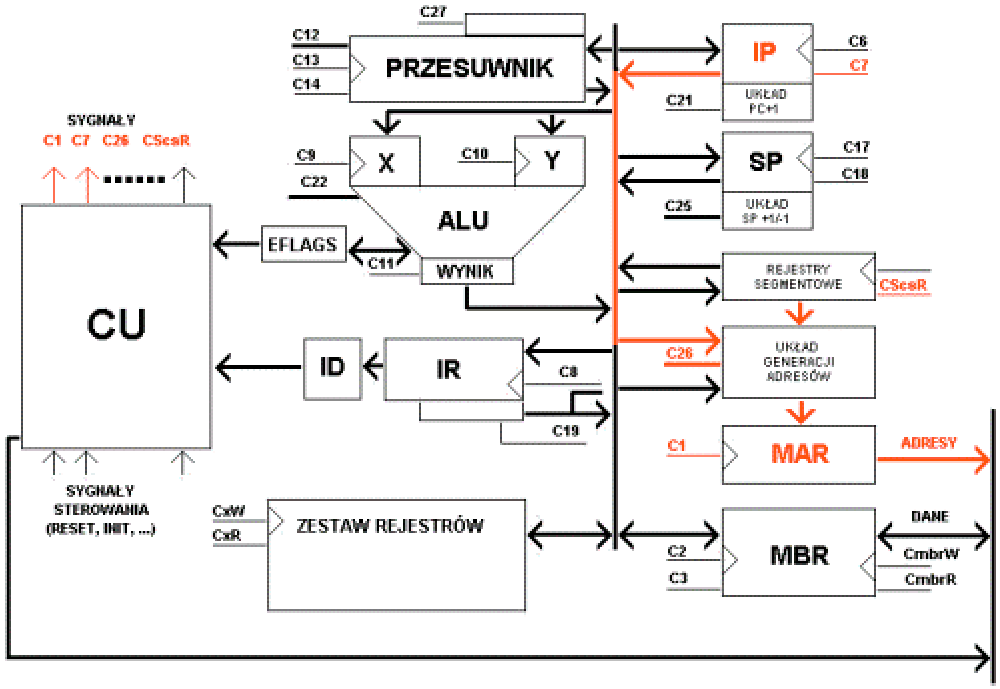
12. FAZA WYKONYWANIA ROZKAZU: Zapis do pamięci. Zawartość rejestru buforowego pamięci MBR zostaje zapisana pod adresem zawartym w rejestrze adresowym MAR



Rozkaz:
ADD [arg. zewnętrzny], AX

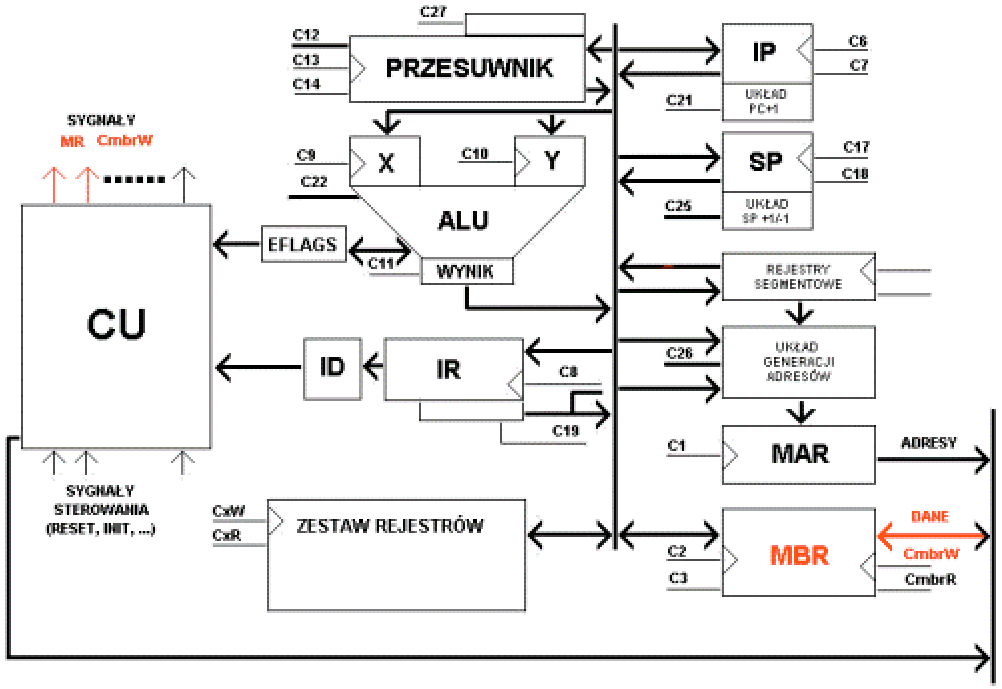
Rozkaz:
ADD [BX+arg. zewnętrzny], AX

5. FAZA WYKONYWANIA ROZKAZU: Wstawienie do rejestru adresowego pamięci MAR adresu znajdującego się w pamięci programu bezpośrednio za kodem rozkazu.



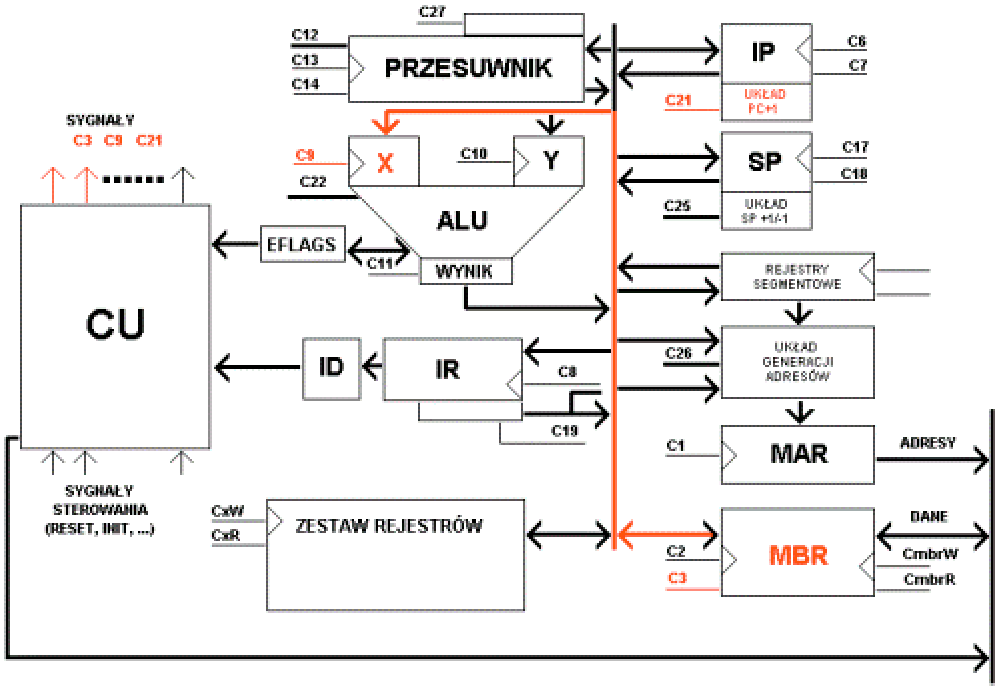
Rozkaz:
ADD [BX+arg. zewnętrzny], AX

6. FAZA WYKONYWANIA ROZKAZU: Odczyt pamięci. Odczytana wartość zostaje zapisana w rejestrze buforowym pamięci MBR.



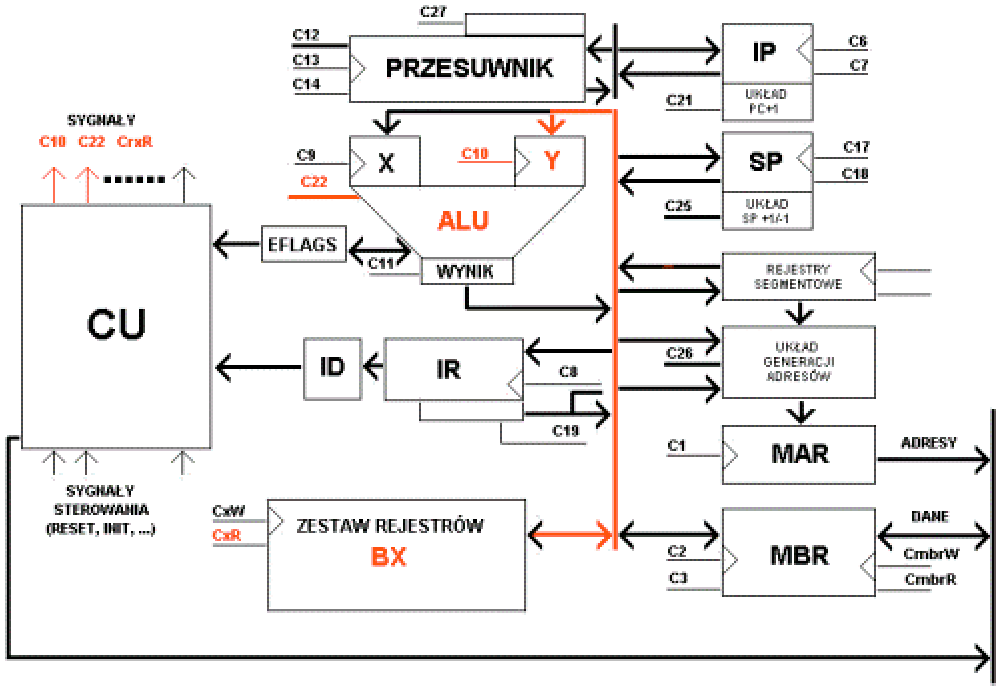
Rozkaz:
ADD [BX+arg. zewnętrzny], AX

7. FAZA WYKONYWANIA ROZKAZU: Przesłanie adresu zawartego w rejestrze buforowego pamięci MBR do rejestru X jednostki ALU.



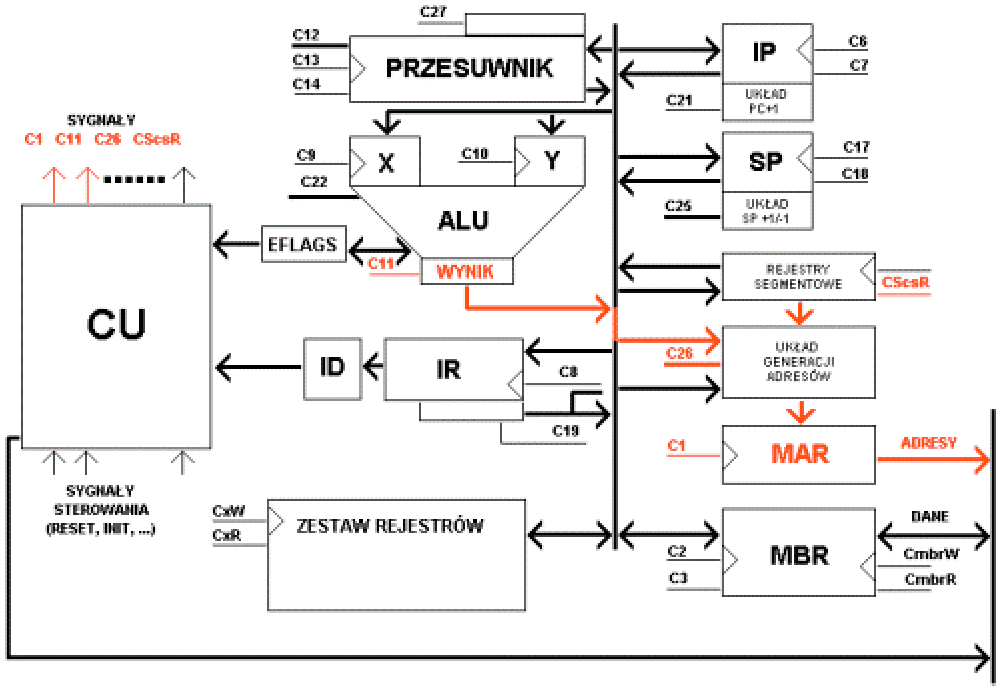
Rozkaz:
ADD [BX+arg. zewnętrzny], AX

8. FAZA WYKONYWANIA ROZKAZU: Przesłanie adresu zawartego w rejestrze BX do rejestru Y jednostki ALU. Wykonanie operacji dodawania.



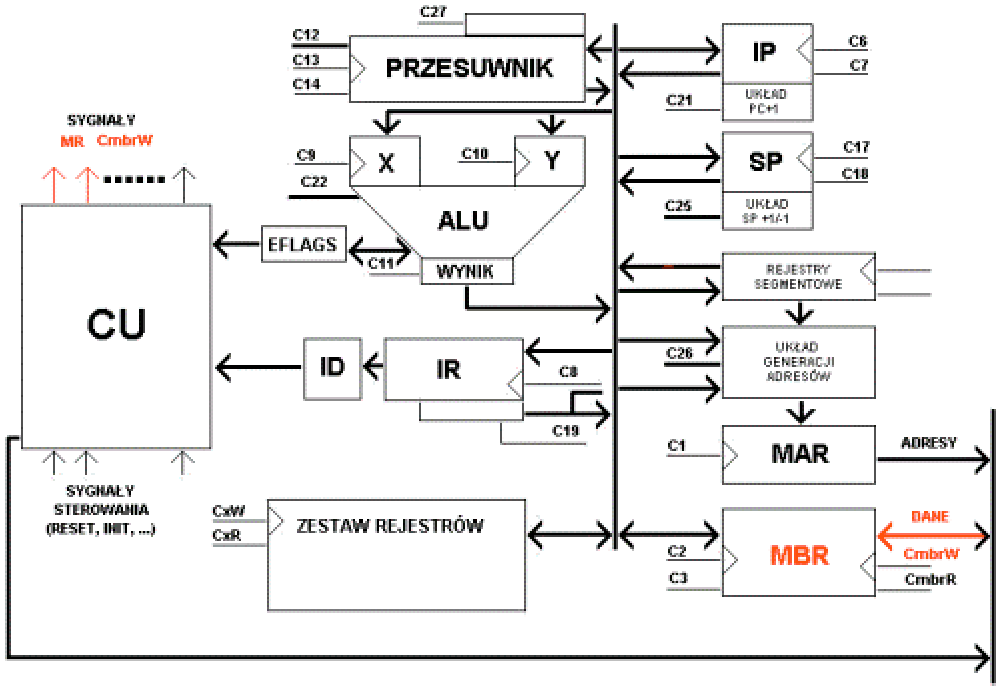
Rozkaz:
ADD [BX+arg. zewnętrzny], AX

9. FAZA WYKONYWANIA ROZKAZU: Przesłanie wyniku dodawania z rejestru wynikowego jednostki ALU do rejestru adresowego pamięci MAR.



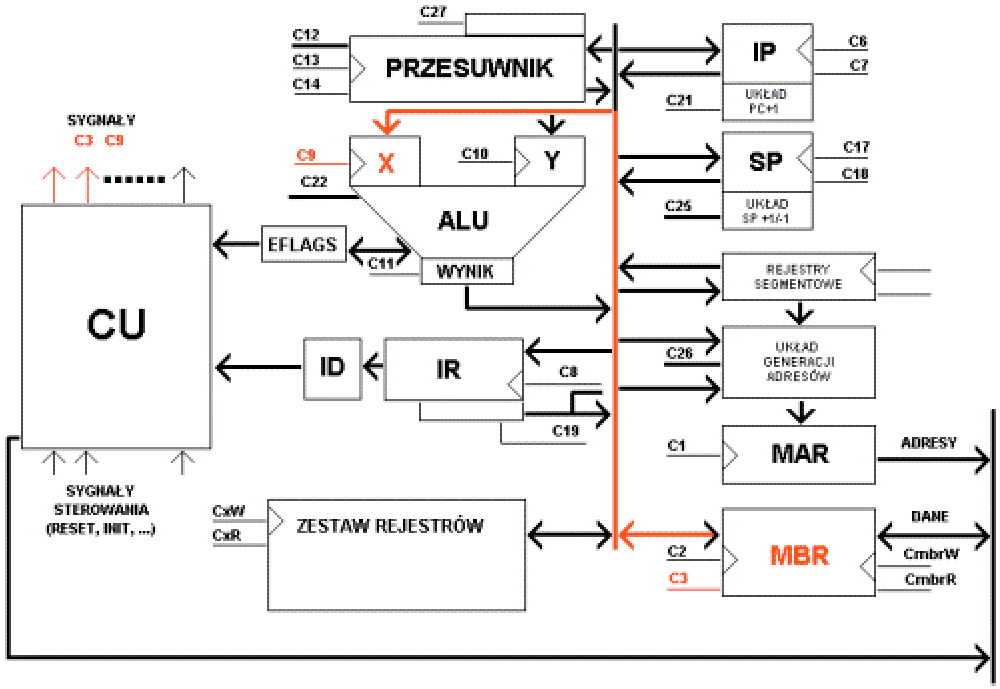
Rozkaz:
ADD [BX+arg. zewnętrzny], AX

10. FAZA WYKONYWANIA ROZKAZU: Odczyt pamięci. Odczytana wartość (w tym przypadku jeden z argumentów rozkazu) zostaje zapisana w rejestrze buforowym pamięci MBR.



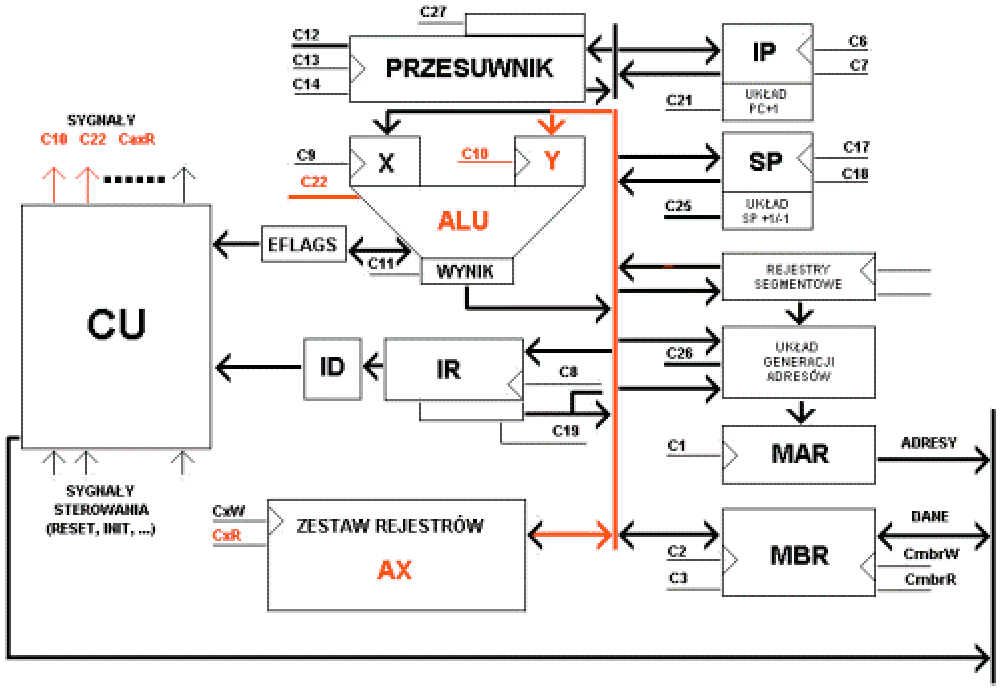
Rozkaz:
ADD [BX+arg. zewnętrzny], AX

11. FAZA WYKONYWANIA ROZKAZU: Przesłanie argumentu rozkazu zawartego w rejestrze buforowym pamięci MBR do rejestru X układu ALU.



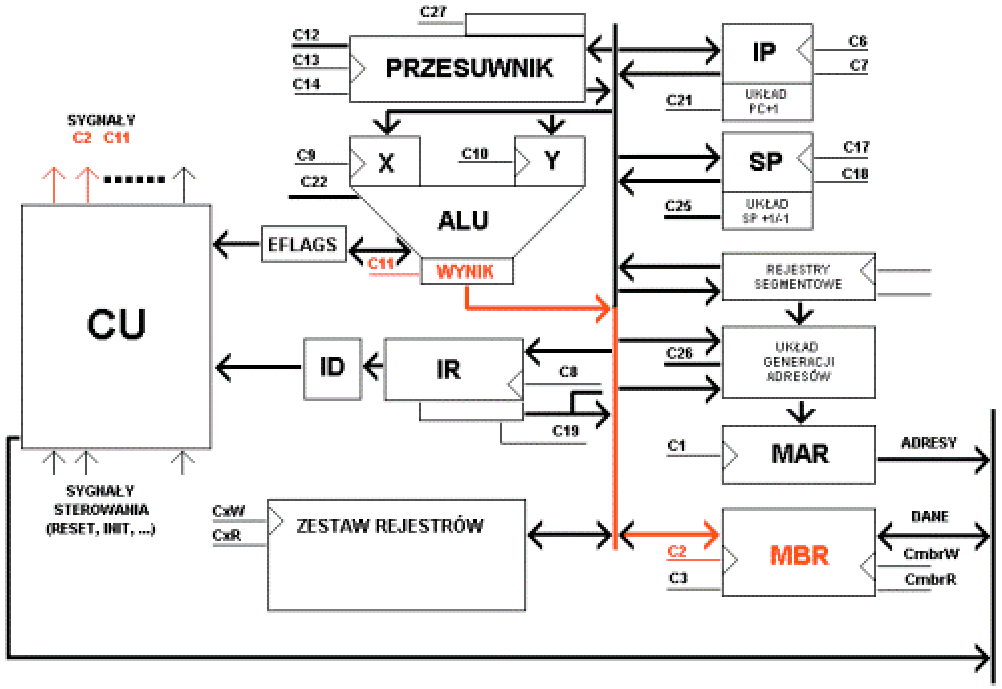
Rozkaz:
ADD [BX+arg. zewnętrzny], AX

12. FAZA WYKONYWANIA ROZKAZU: Przesłanie drugiego argumentu rozkazu z rejestru AX do rejestru Y jednostki ALU. Wykonanie operacji dodawania.



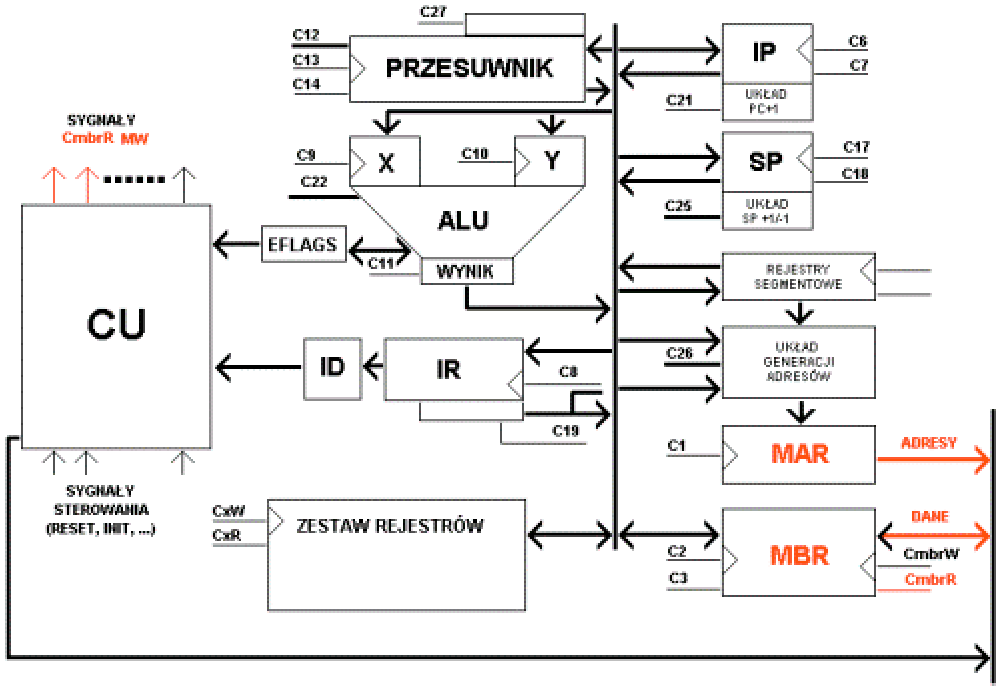
Rozkaz:
ADD [BX+arg. zewnętrzny], AX

13. FAZA WYKONYWANIA ROZKAZU: Przesłanie wyniku operacji dodawania z rejestru wynikowego jednostki ALU do rejestru buforowego pamięci MBR



Rozkaz:
ADD [BX+arg. zewnętrzny], AX

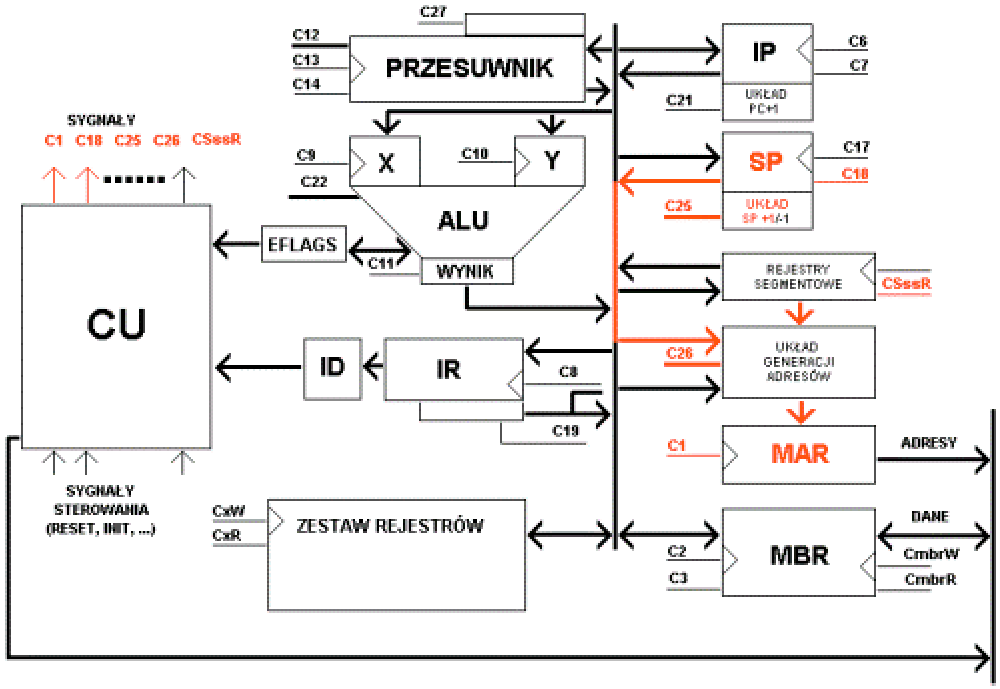
14. FAZA WYKONYWANIA ROZKAZU: Zapis do pamięci. Zawartość rejestru buforowego pamięci MBR zostaje zapisana pod adresem zawartym w rejestrze adresowym MAR



Rozkaz:
ADD [BX+arg. zewnętrzny], AX

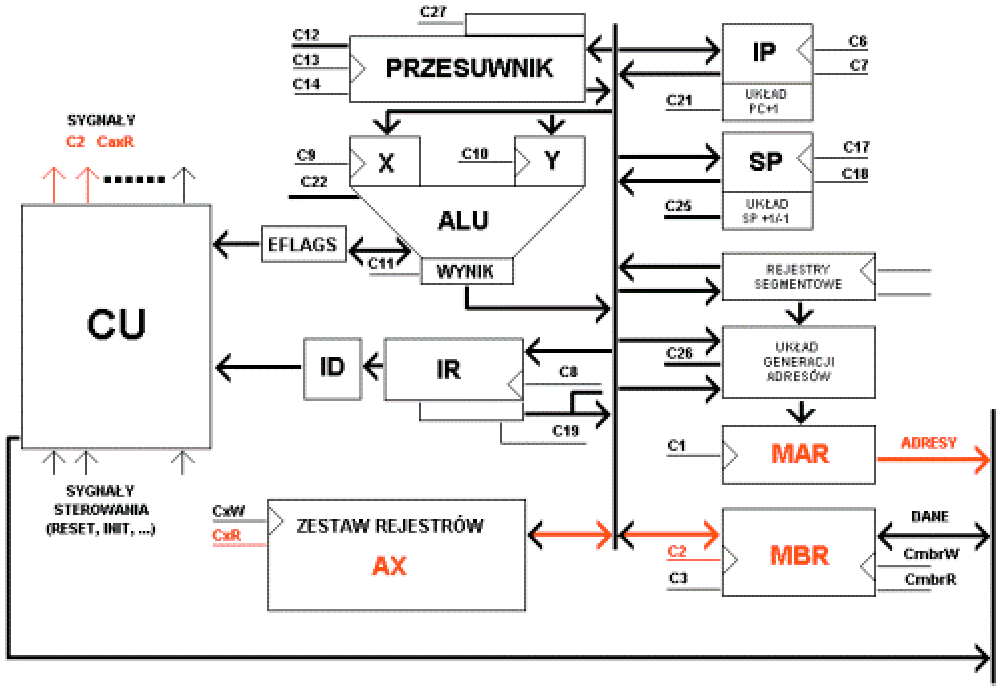
Rozkaz:
PUSH AX

5. FAZA WYKONYWANIA ROZKAZU: Wysłanie do rejestru adresowego pamięci MAR adresu zawartego w rejestrze wskaźnika stosu SP. Inkrementacja zawartości rejestru wskaźnika stosu SP.



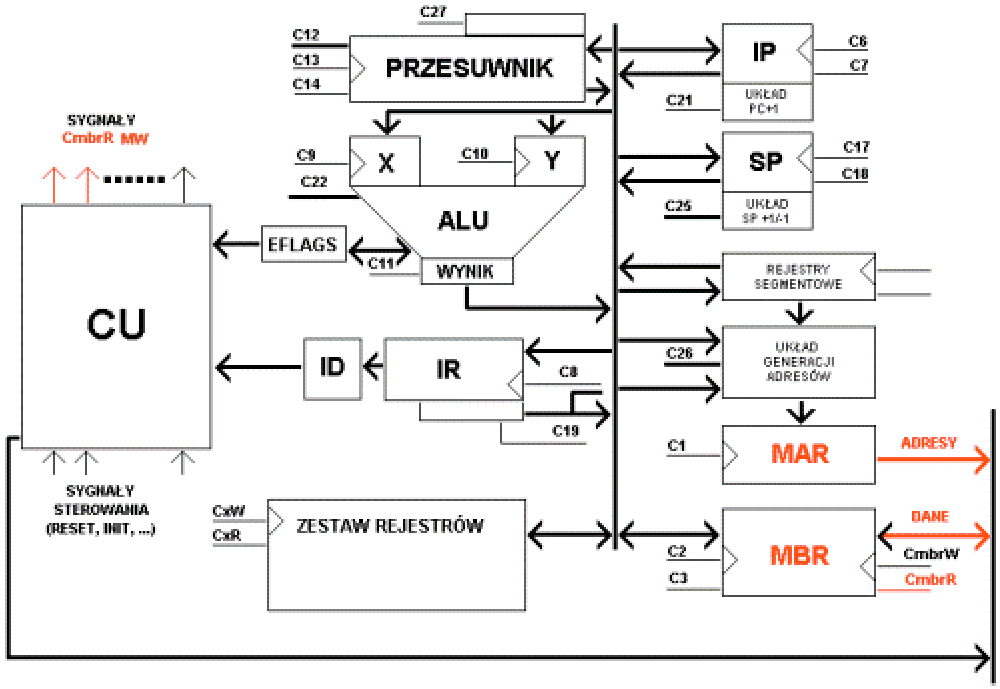
Rozkaz:
PUSH AX

6. FAZA WYKONYWANIA ROZKAZU: Zapis do rejestru buforowego pamięci MBR zawartości rejestru AX.



Rozkaz:
PUSH AX

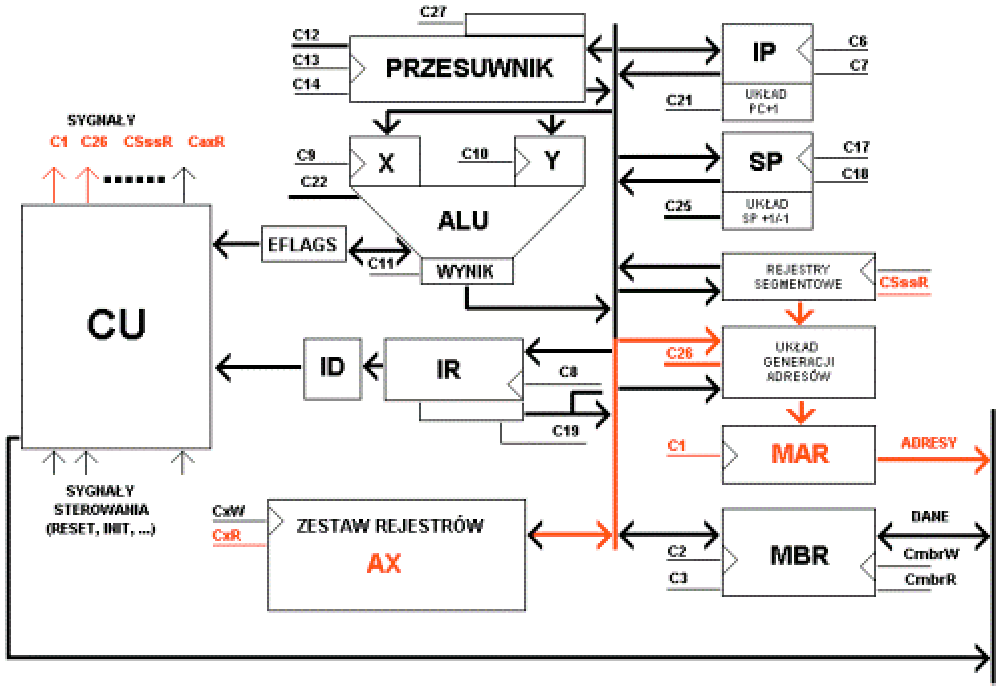
7. FAZA WYKONYWANIA ROZKAZU: Zapis do pamięci. Zawartość rejestru buforowego pamięci MBR zostaje zapisana pod adresem zawartym w rejestrze adresowym MAR



Rozkaz:
PUSH AX

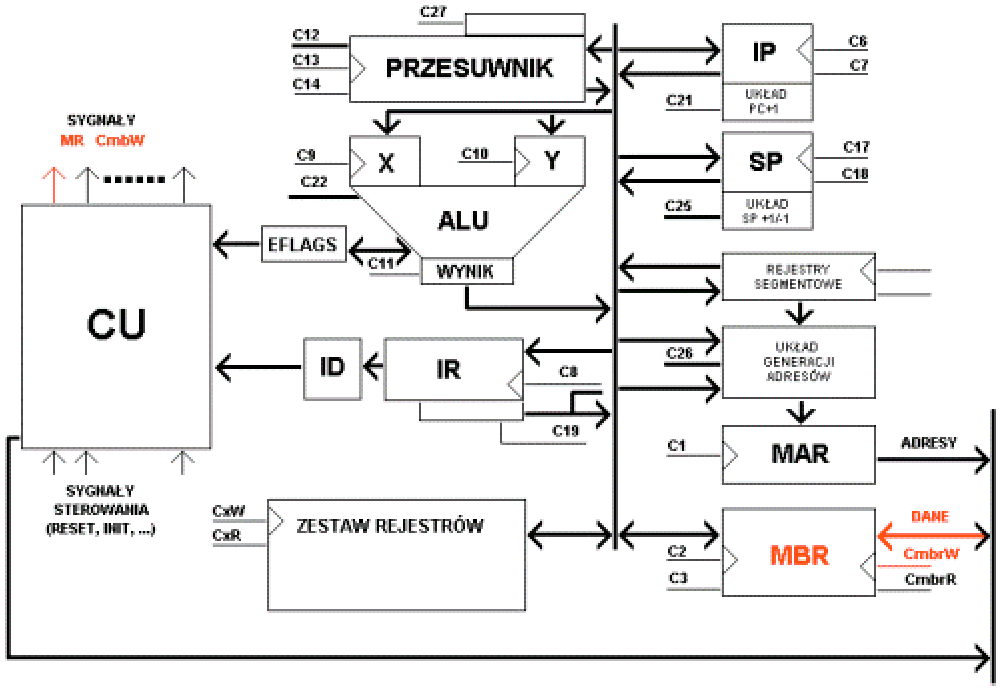
Rozkaz:
PUSH [AX]

5. FAZA WYKONYWANIA ROZKAZU: Wysłanie do rejestry adresowego pamięci MAR zawartości rejestru AX.



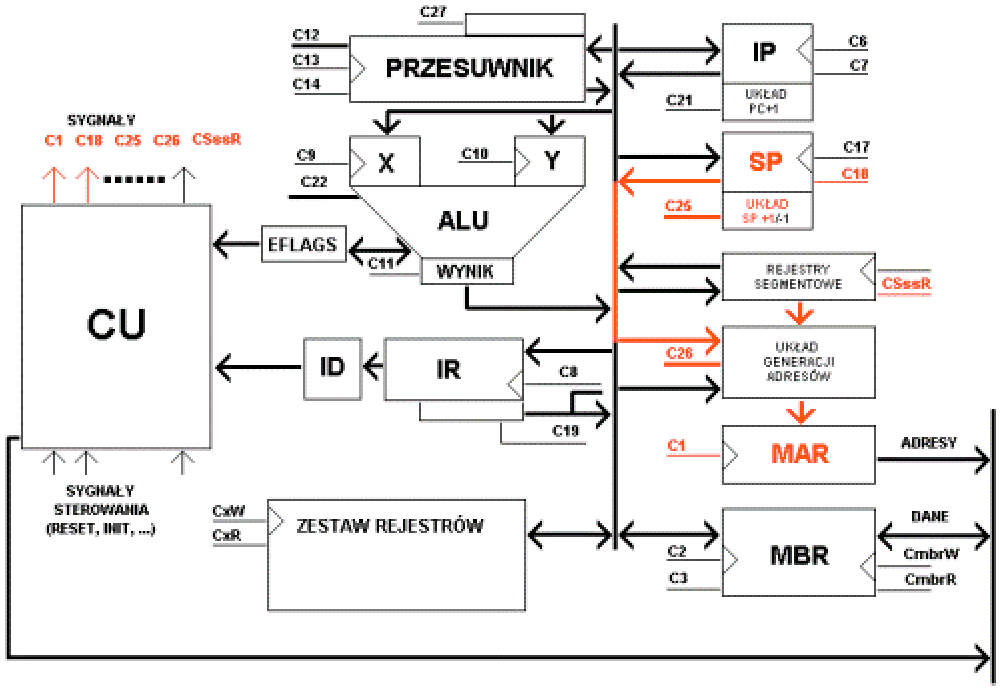
Rozkaz:
PUSH [AX]

6. FAZA WYKONYWANIA ROZKAZU: Odczyt pamięci. Odczytana wartość (w tym przypadku argument rozkazu) zostaje zapisana w rejestrze buforowym pamięci MBR.



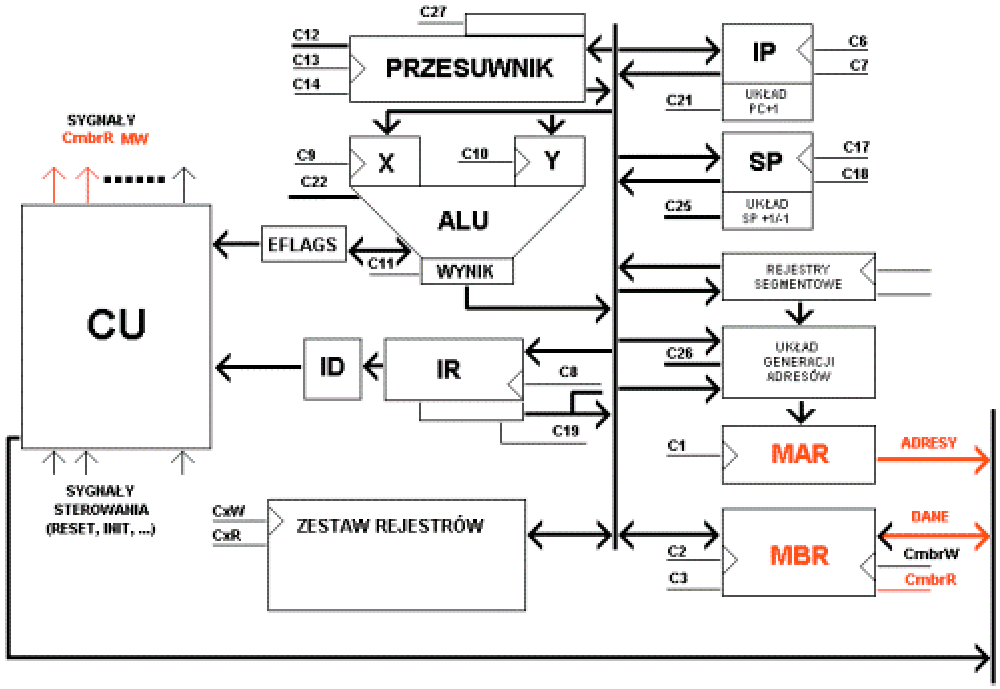
Rozkaz:
PUSH [AX]

7. FAZA WYKONYWANIA ROZKAZU: Wysłanie do rejestru adresowego pamięci MAR adresu zawartego w rejestrze wskaźnika stosu SP. Inkrementacja zawartości rejestru wskaźnika stosu SP.



Rozkaz:
PUSH [AX]

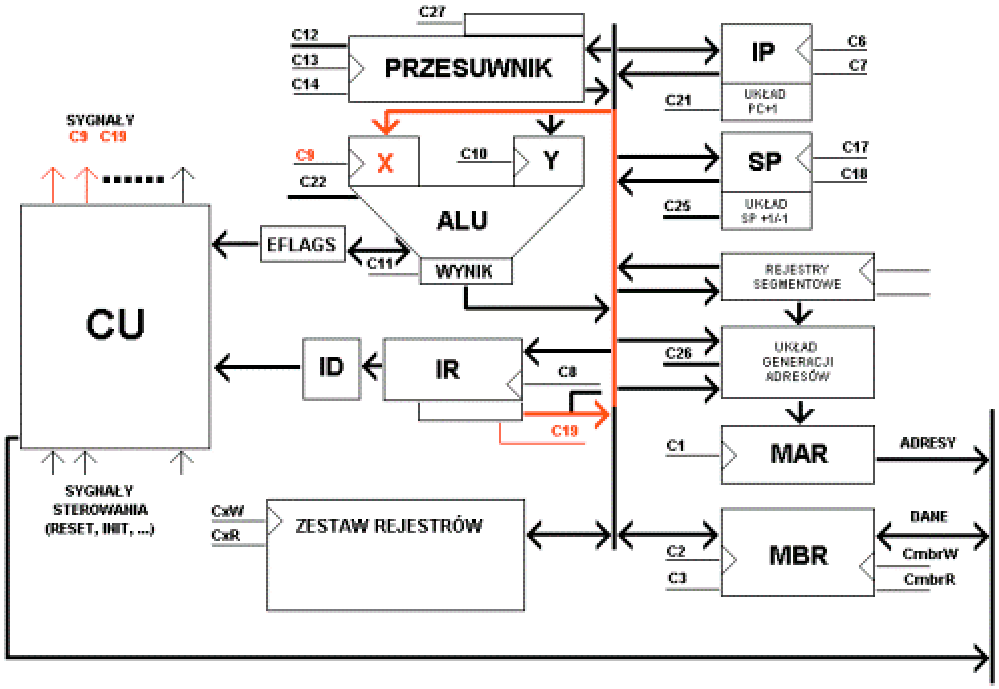
8. FAZA WYKONYWANIA ROZKAZU: Zapis do pamięci. Zawartość rejestru buforowego pamięci MBR zostaje zapisana pod adresem zawartym w rejestrze adresowym MAR



Rozkaz:
PUSH [AX]

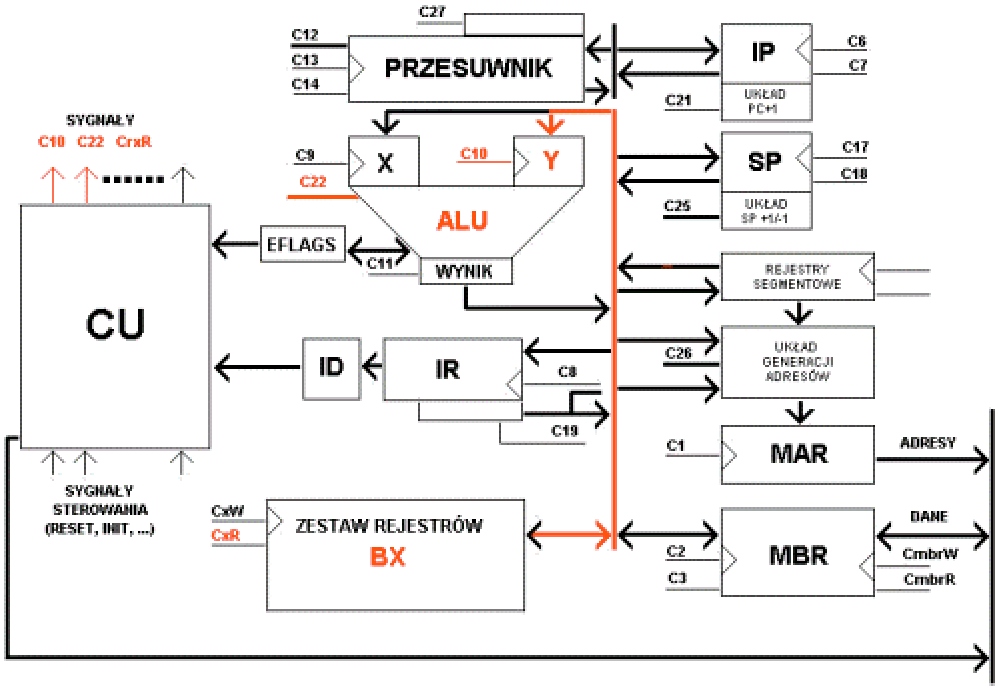
Rozkaz:
SUB [BX+arg. wbudowany], AX

5. FAZA WYKONYWANIA ROZKAZU: Wstawienie do rejestru X jednostki ALU zawartości pola natychmiastowego rejestru rozkazu IR.



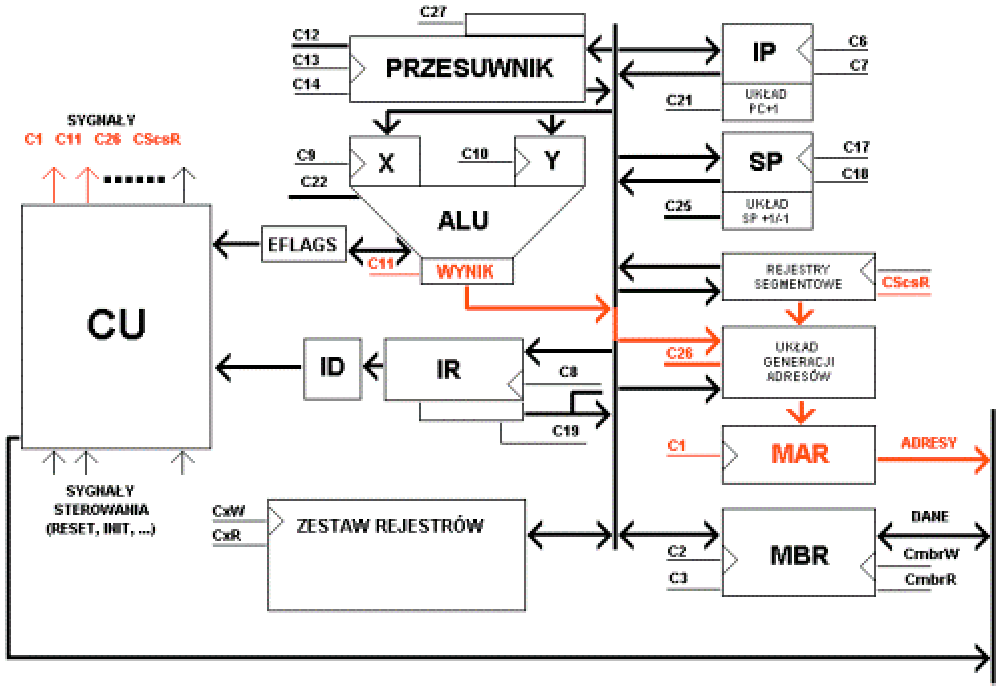
Rozkaz:
SUB [BX+arg. wbudowany], AX

6. FAZA WYKONYWANIA ROZKAZU: Wstawienie do rejestru Y jednostki ALU zawartości rejestru BX. Wykonanie operacji dodawania.



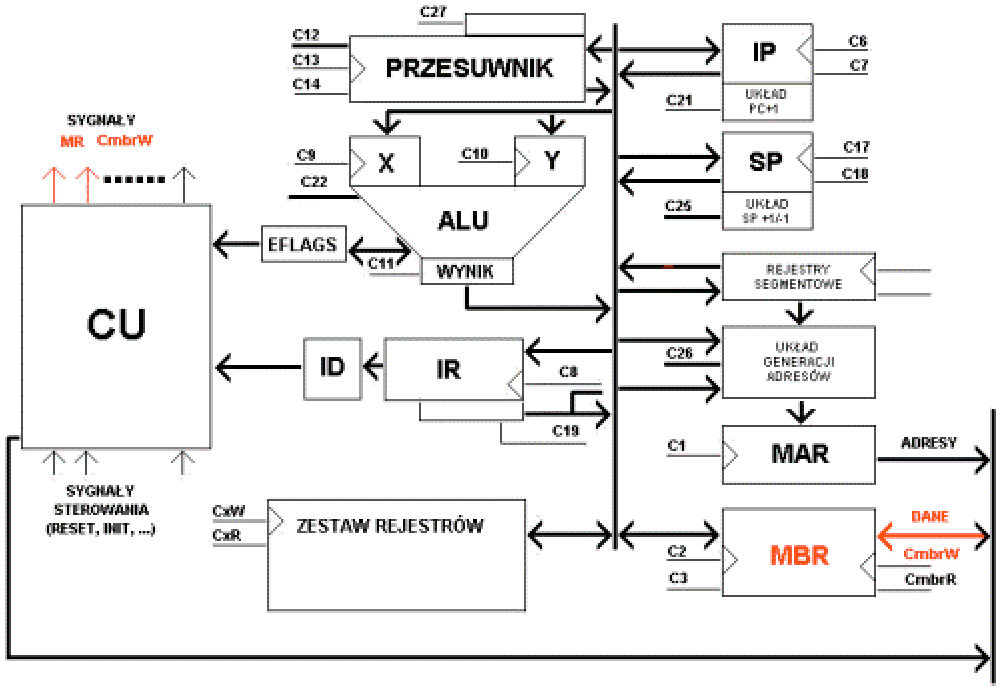
Rozkaz:
SUB [BX+arg. wbudowany], AX

7. FAZA WYKONYWANIA ROZKAZU: Przesłanie wyniku dodawania z rejestru wynikowego jednostki ALU do rejestru adresowego pamięci MAR.



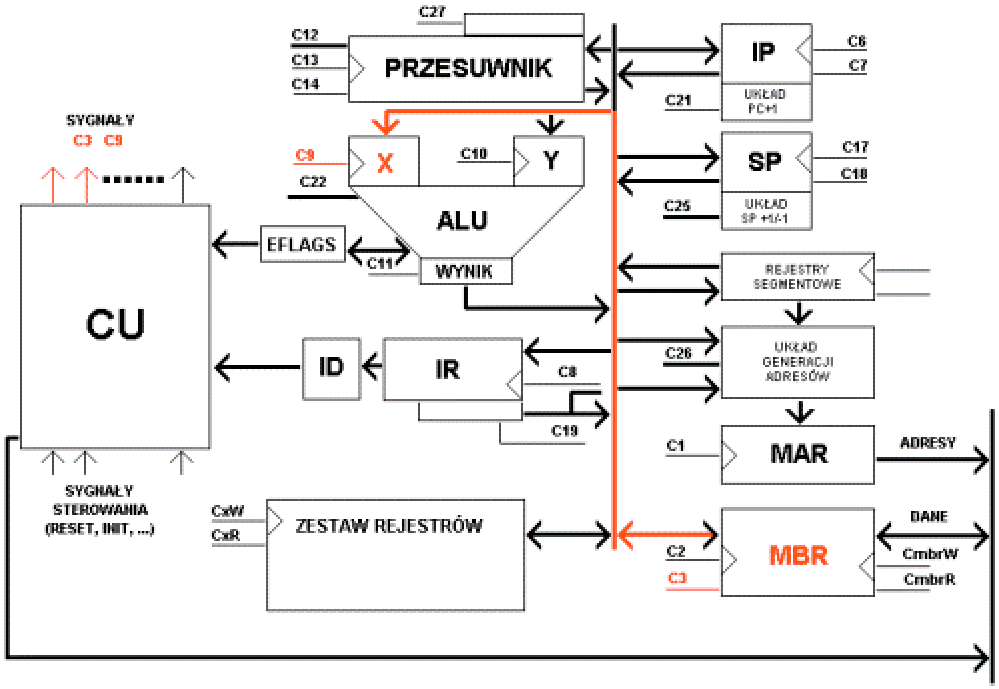
Rozkaz:
SUB [BX+arg. wbudowany], AX

8. FAZA WYKONYWANIA ROZKAZU: Odczyt pamięci. Odczytana wartość (w tym przypadku jeden z argumentów rozkazu) zostaje zapisana w rejestrze buforowym pamięci MBR.



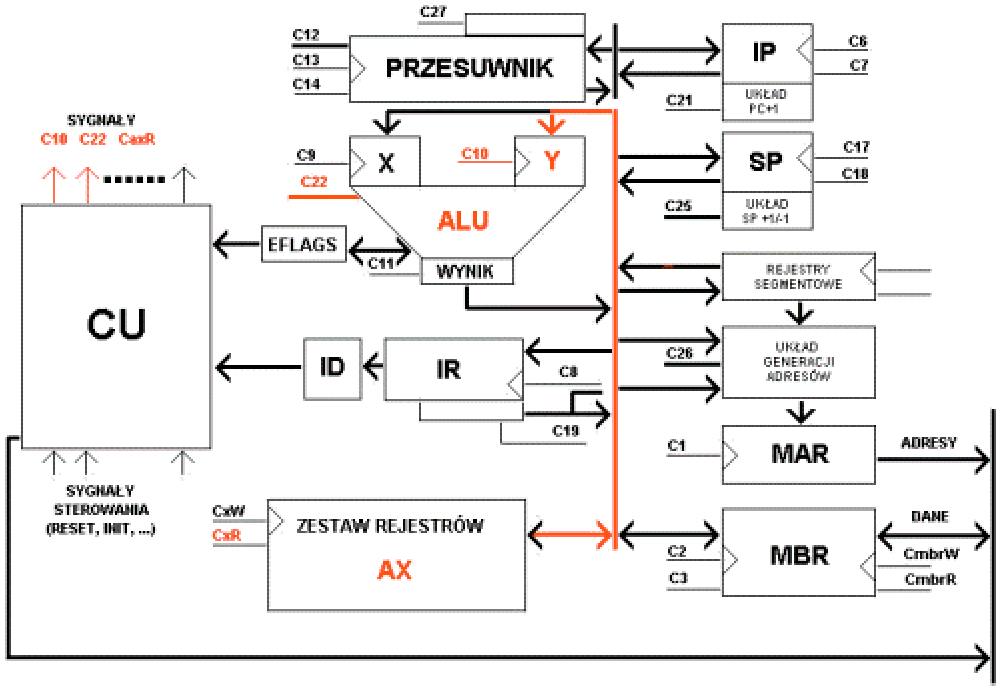
Rozkaz:
SUB [BX+arg. wbudowany], AX

9. FAZA WYKONYWANIA ROZKAZU: Przesłanie argumentu rozkazu zawartego w rejestrze buforowym pamięci MBR do rejestru X układu ALU.



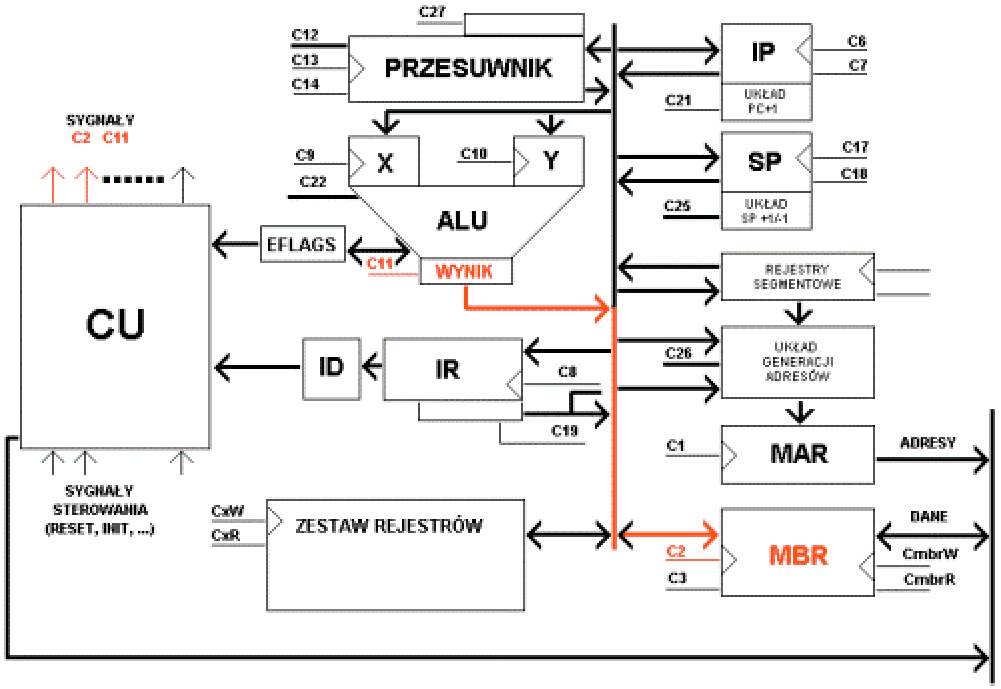
Rozkaz:
SUB [BX+arg. wbudowany], AX

10. FAZA WYKONYWANIA ROZKAZU: Przesłanie drugiego argumentu rozkazu z rejestru AX do rejestru Y jednostki ALU. Wykonanie operacji odejmowania.



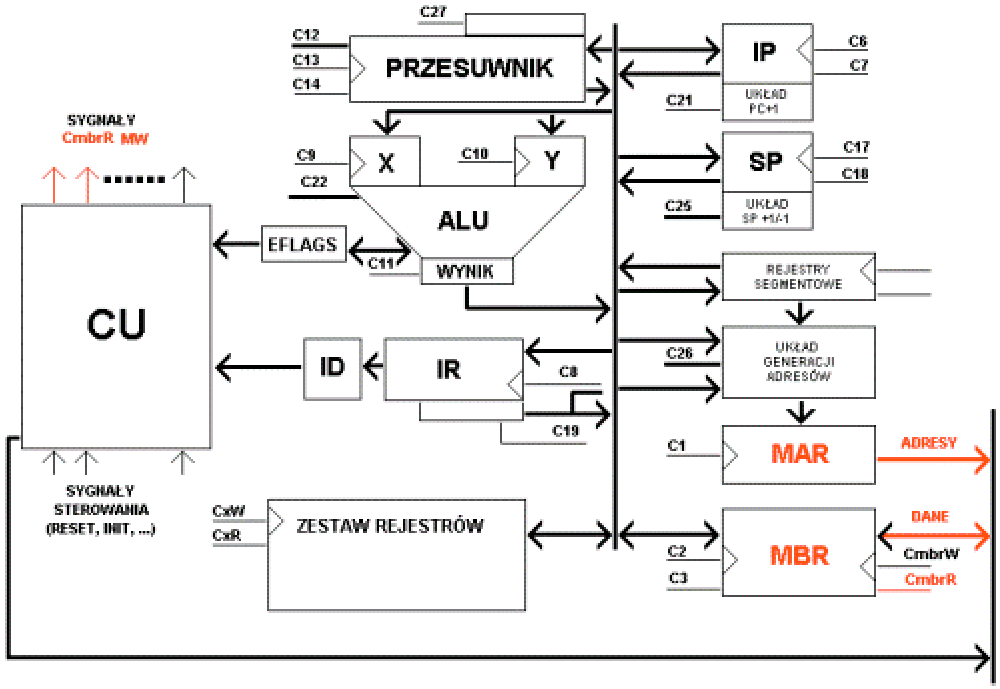
Rozkaz:
SUB [BX+arg. wbudowany], AX

11. FAZA WYKONYWANIA ROZKAZU: Przesłanie wyniku operacji odejmowania z rejestru wynikowego jednostki ALU do rejestru buforowego pamięci MBR



Rozkaz:
SUB [BX+arg. wbudowany], AX

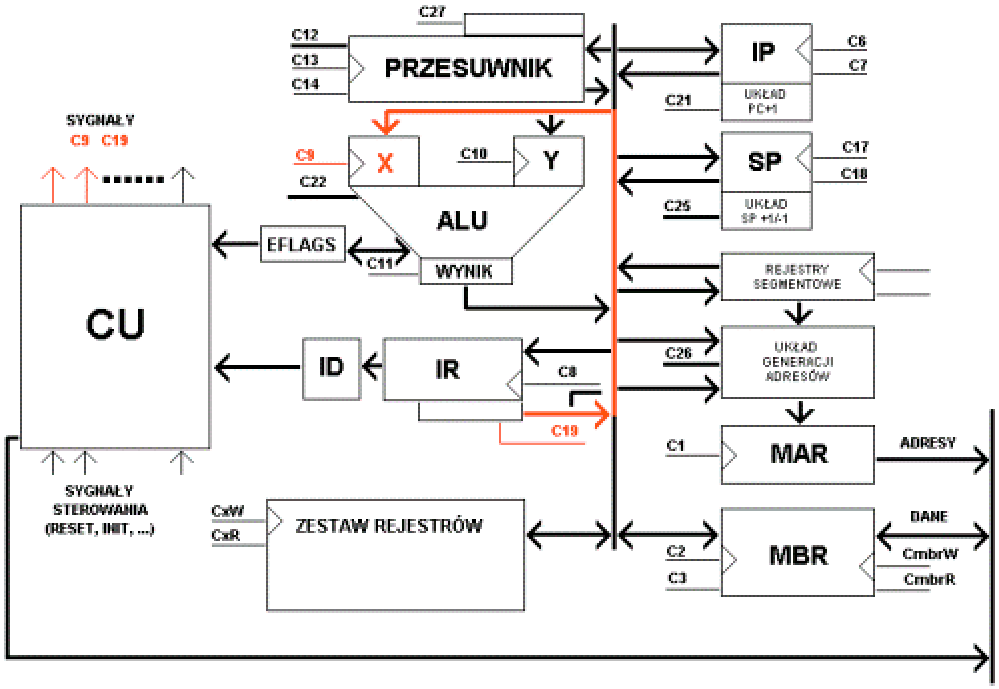
12. FAZA WYKONYWANIA ROZKAZU: Zapis do pamięci. Zawartość rejestru buforowego pamięci MBR zostaje zapisana pod adresem zawartym w rejestrze adresowym MAR



Rozkaz:
SUB [BX+arg. wbudowany], AX

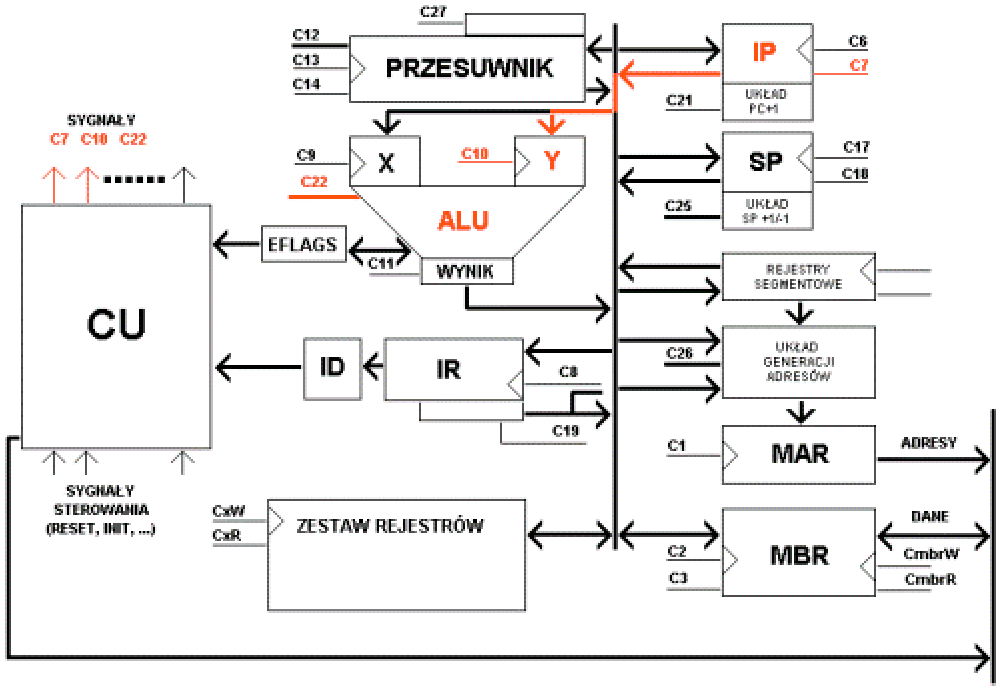
Rozkaz:
JZ PC+arg. wbudowany

5. FAZA POBRANIA ROZKAZU: Wstawienie do rejestru X jednostki ALU zawartości pola natychmiastowego rejestru rozkazu IR.



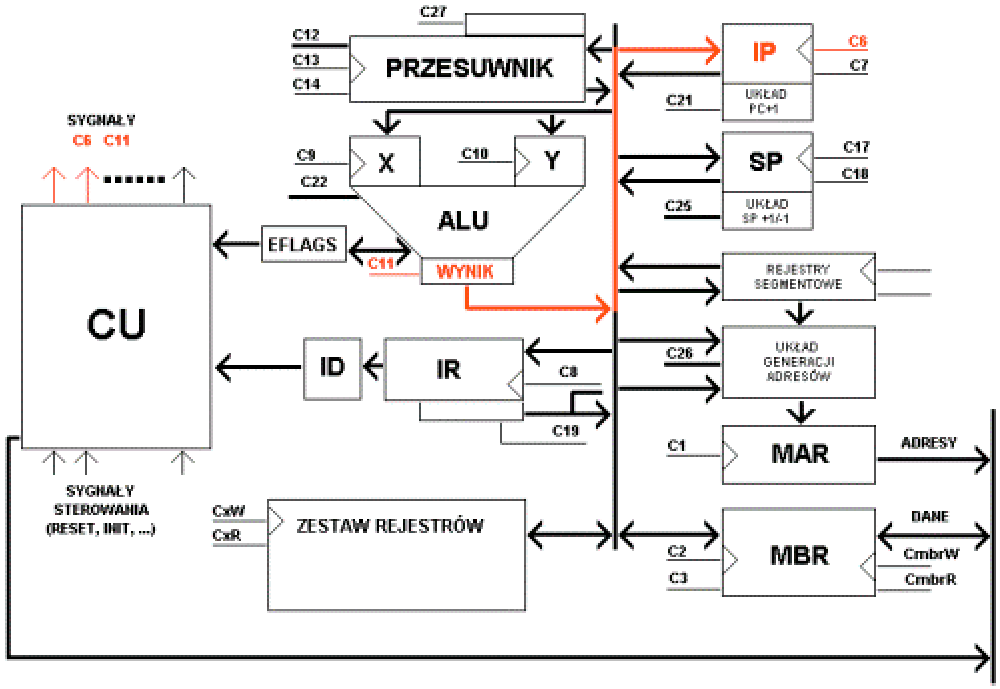
Rozkaz:
JZ PC+arg. wbudowany

6. FAZA POBRANIA ROZKAZU: Wstawienie do rejestru Y jednostki ALU zawartości licznika programu IP. Wykonanie operacji dodawania.



Rozkaz:
JZ PC+arg. wbudowany

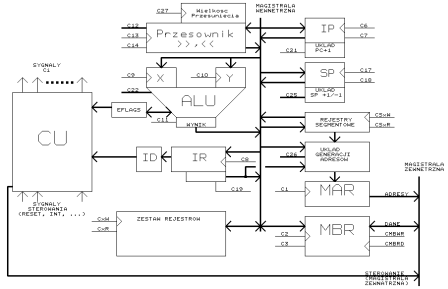
7. FAZA POBRANIA ROZKAZU: Przesłanie wyniku dodawania z rejestru wynikowego jednostki ALU do licznika programu IP.

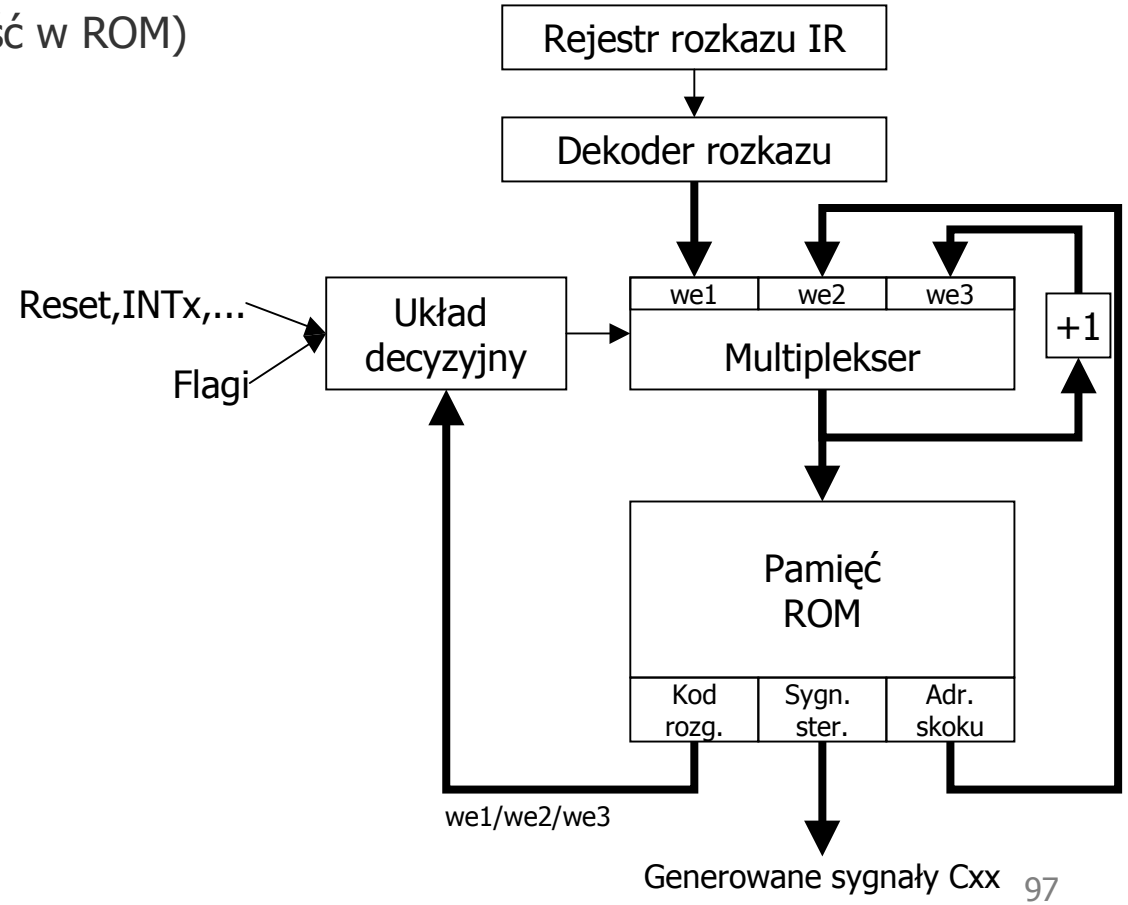


Rozkaz:
JZ PC+arg. wbudowany

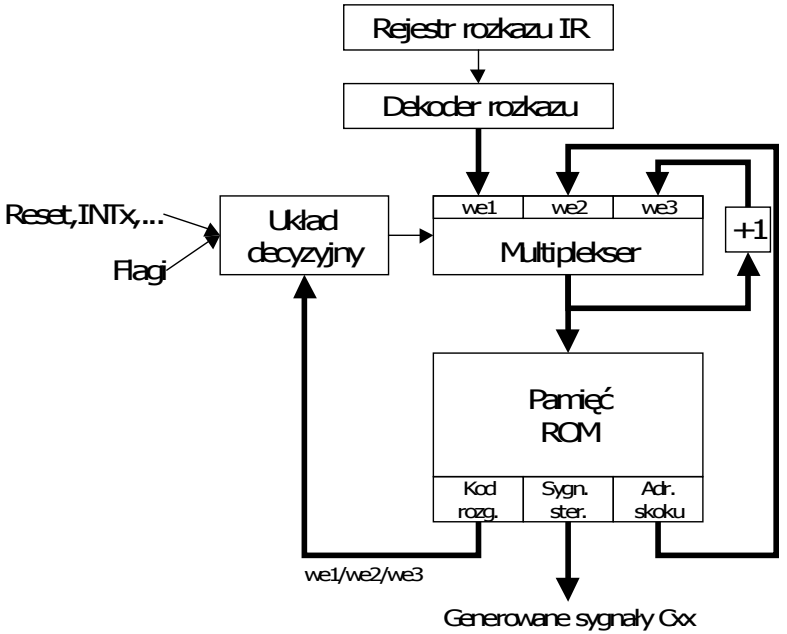
Realizacja jednostki sterującej

- **Jednostka sterująca (CU)**
  - Odpowiada za wytworzenie sygnałów sterujących
  - Podczas pracy bazuje na:
    - sygnałach: zegar, Reset, przerwania, ...
    - flagi
    - swoim mikro-kodzie (treść w ROM)



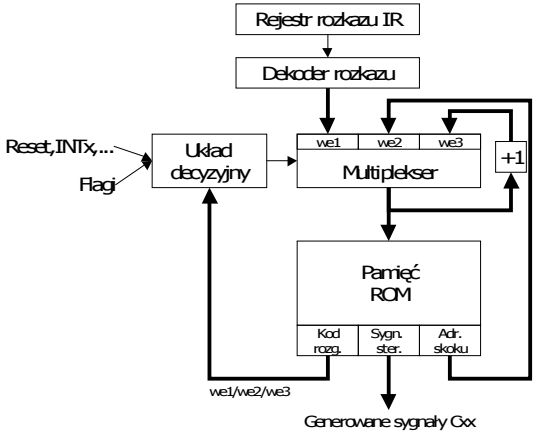


- **Jednostka sterująca (CU)**
  - Treść mikro-kodu (pamięci ROM w CU)
    - Wybór wejścia przez multiplekser
      - wej. 1 - z dekodera rozkazów
      - wej. 2 - z pola adresu skoku
      - wej. 3 - z następnika (+1)
    - CU zawsze po RESET startuje od adresu 0




```
Adres    Skok     Sygnały ster.     Mux-op
0        -        C7, C26, C1       +1       //początek fazy pobrania
1        -        CMBWR             +1
2        ID       C3, C8, C21(+)    „WE1”    //koniec fazy pobrania i dek. inst.
...
100      -        C19, C9           +1       //JMP PC+arg.wew
101      -        C7, C10, C22(+)   +1
102      0        C11, C6           „skok”   //koniec fazy wykonania JMP...
...
200      -        CaR, C9           +1       //ADD A, B
201      -        CbR, C10, C22(+)  +1
202      0        C11, CaW          „skok”   //koniec fazy wykonania ADD...
...
```

- **Jednostka sterująca (CU)**
  - Co do pamięci CU wprowadzają przerwania?




```
Adres    Skok     Sygnały ster.     Mux-op
0        -        C7, C26, C1       +1       //początek fazy pobrania
1        -        CMBWR             +1
2        ID       C3, C8, C21(+)    -        //koniec fazy pobrania i dek. inst.
...
100      -        C19, C9           +1       //JMP PC+arg.wew
101      -        C7, C10, C22(+)   +1
102      998      C11, C6           „skok”   //koniec fazy wykonania JMP...
...
200      -        CaR, C9           +1       //ADD A, B
201      -        CbR, C10, C22(+)  +1
202      998      C11, CaW          „skok”   //koniec fazy wykonania ADD...
...
998      ...      ...               „+1”     //warunkowa obsługa przerwania (IF)
999      0        ...               „skok”   //...
```

Dziękuje za uwagę