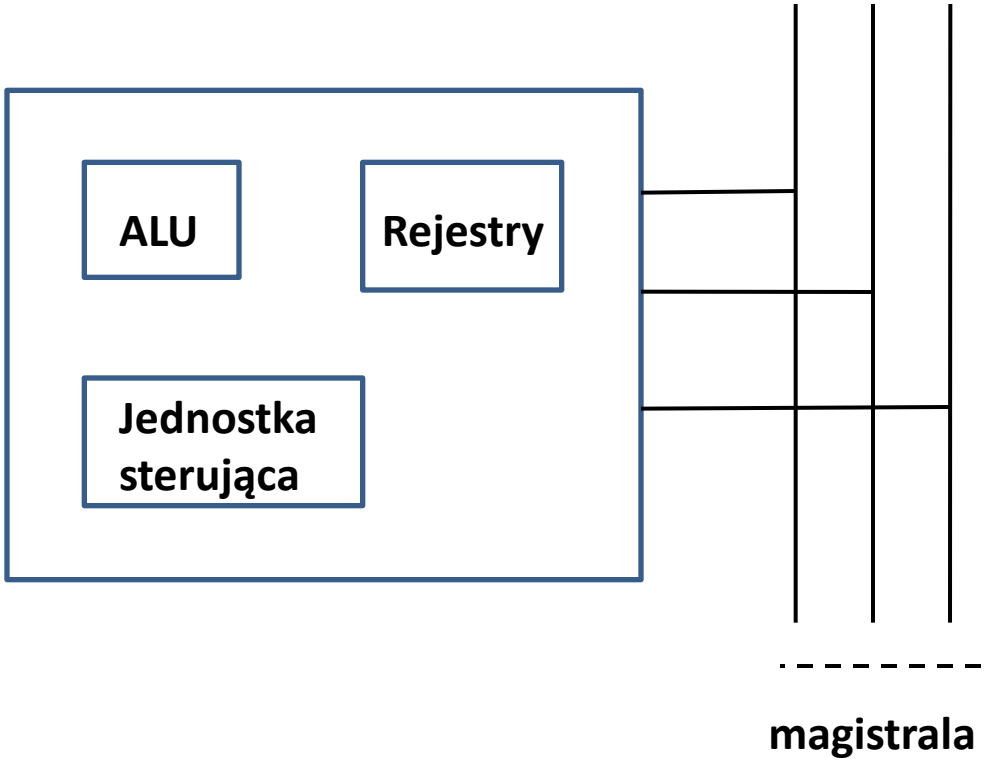
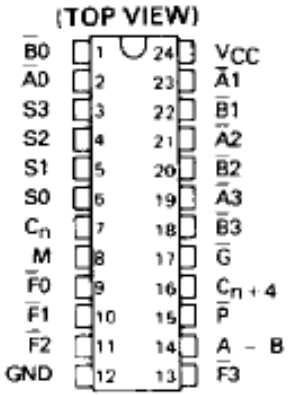
Struktura i działanie jednostki centralnej



Zadania procesora:

- **Pobieranie rozkazów;**
- **Interpretowanie rozkazów;**
- **Pobieranie danych**
- **Przetwarzanie danych**
- **Zapisywanie danych**

Blok arytmetyczno-logiczny - ALU



- Wykonuje operacje na 4-bitowych zmiennych **A** i **B**. Wynik umieszczany jest w **F**
- Wejścia **S** i **M** – służą do programowania funkcji wykonywanych przez układ
- **Cn** – przeniesienie wejściowe
- **Cn+4** - przeniesienie wyjściowe

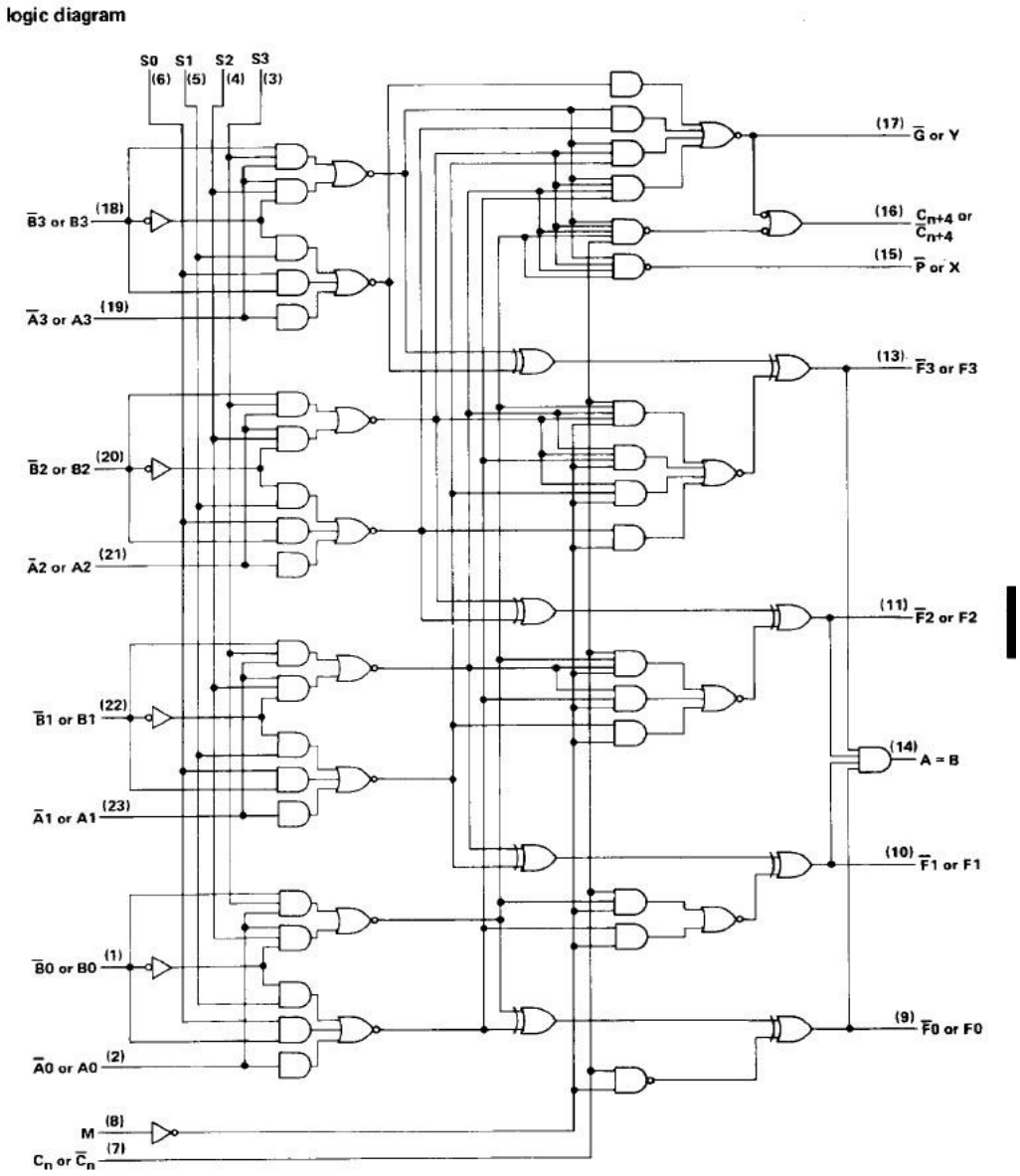
SN 74181

Blok arytmetyczno-logiczny - ALU

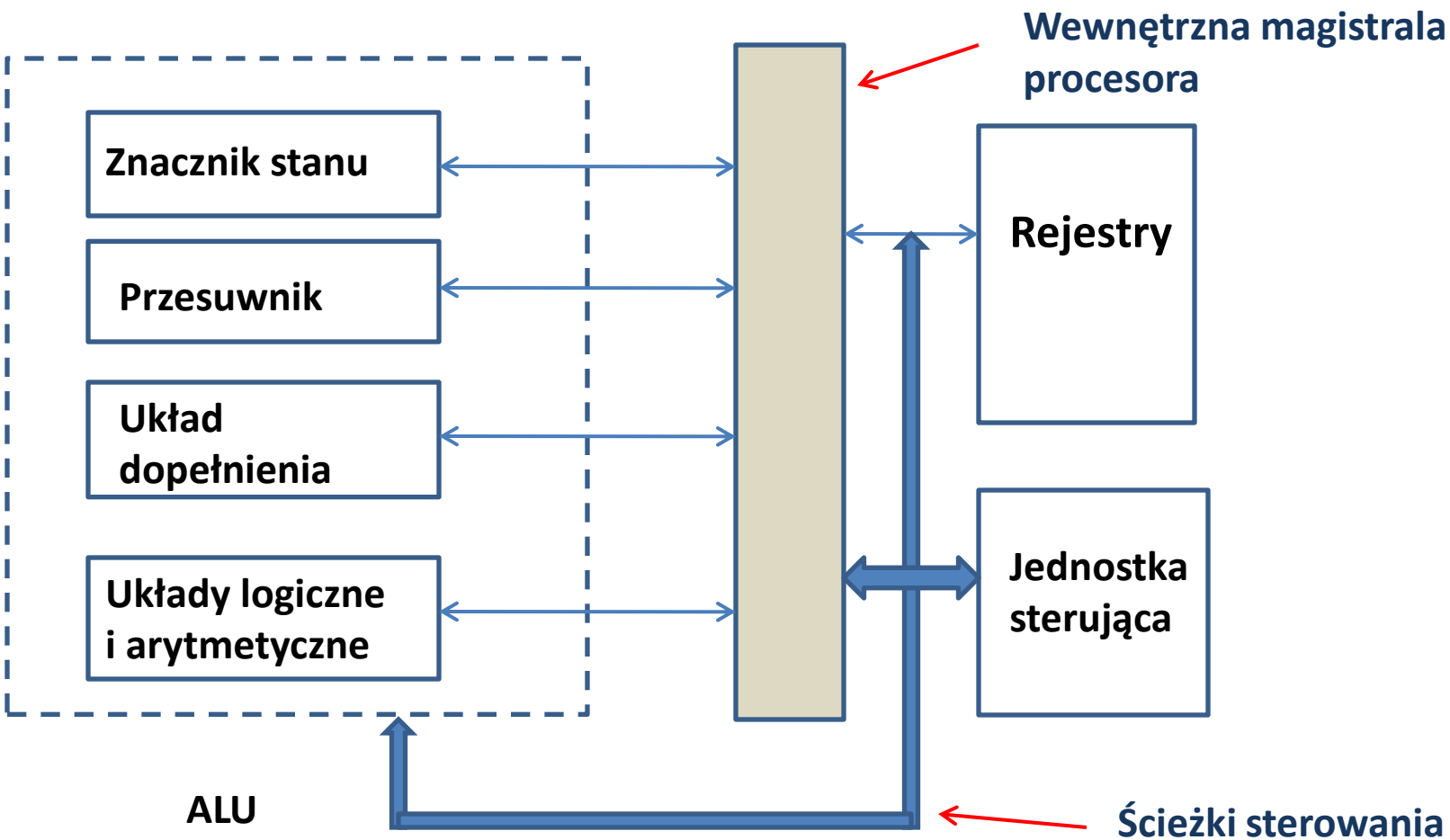
Realizowane funkcje:

SELECTION				ACTIVE-HIGH DATA		
				M = H	M = L; ARITHMETIC OPERATIONS	
S3	S2	S1	S0	LOGIC FUNCTIONS	$\overline{C_n}$ = H (no carry)	$\overline{C_n}$ = L (with carry)
L	L	L	L	$F = \overline{A}$	F = A	F = A PLUS 1
L	L	L	H	$F = \overline{A + B}$	F = A + B	F = (A + B) PLUS 1
L	L	H	L	$F = \overline{A}B$	$F = A + \overline{B}$	$F = (A + \overline{B})$ PLUS 1
L	L	H	H	F = 0	F = MINUS 1 (2's COMPL)	F = ZERO
L	H	L	L	$F = \overline{AB}$	$F = A$ PLUS $A\overline{B}$	$F = A$ PLUS $A\overline{B}$ PLUS 1
L	H	L	H	$F = \overline{B}$	$F = (A + B)$ PLUS $A\overline{B}$	$F = (A + B)$ PLUS $A\overline{B}$ PLUS 1
L	H	H	L	$F = A \oplus B$	F = A MINUS B MINUS 1	F = A MINUS B
L	H	H	H	$F = A\overline{B}$	$F = A\overline{B}$ MINUS 1	$F = A\overline{B}$
H	L	L	L	$F = \overline{A} + B$	F = A PLUS AB	F = A PLUS AB PLUS 1
H	L	L	H	$F = \overline{A \oplus B}$	F = A PLUS B	F = A PLUS B PLUS 1
H	L	H	L	F = B	$F = (A + \overline{B})$ PLUS AB	$F = (A + \overline{B})$ PLUS AB PLUS 1
H	L	H	H	F = AB	F = AB MINUS 1	F = AB
H	H	L	L	F = 1	F = A	F = A PLUS A PLUS 1
H	H	L	H	$F = A + \overline{B}$	F = (A + B) PLUS A	F = (A + B) PLUS A PLUS 1
H	H	H	L	F = A + B	$F = (A + \overline{B})$ PLUS A	$F = (A + \overline{B})$ PLUS A PLUS 1
H	H	H	H	F = A	F = A MINUS 1	F = A

Blok arytmetyczno-logiczny - ALU



Wewnętrzna struktura procesora



Rejestry można podzielić na:

- **rejestry widzialne dla użytkownika;**
- **rejestry sterowania i stanu – są one używane przez jednostkę sterującą do sterowania pracą procesora oraz przez uprzywilejowane programy systemu operacujnego;**

Rejestry widzialne przez użytkownika:

- **ogólnego przeznaczenia (general purpose);**
- **danych;**
- **adresów;**
- **kodów warunkowych;**

Rejestry adresowe:

- **wskaźnik segmentu;**
- **indeksowe;**
- **wskaźnik stosu – rejstr wskazujący wierzchołek stosu**

Rejestry kodów warunkowych:

- **Kody warunkowe są bitami ustawianymi sprzętowo przez procesor w wyniku operacji . Np. operacja arytmetyczna może dać wynik dodatni, ujemny, zerowy lub przepełnienie.**
- **Poza umieszczeniem wyniku w rejestrze danych lub w pamięci ustalany jest również kod warunkowy.**
- **Kod ten może być następnie sprawdzany w ramach rozgałęzienia warunkowego.**
- **Bity kodu warunkowego zbierane są w jednym lub wielu rejestrach.**

Rejestry sterowania i stanu:

- **licznik programu (PC) – zawiera adres rozkazu przewidzianego do pobrania;**

- **rejestr rozkazu (IR) – zawiera ostatnio pobrany rozkaz;**

- **rejestr adresowy pamięci (MAR) – zawiera adres pamięci, z którego (do którego) ma odczytać (zapisać) słowo danych;**

- **rejestr buforowy pamięci (MBR) – zawiera słowo danych, które ma być zapisane/odczytane z pamięci;**

- **rejestr stanu programu (PSW)**

Rejestr stanu programu (PSW) zawiera następujace pola:

- **znak – bit znaku wyniku ostatniej operacji;**
- **zero – ustawiane, gdy wynik operacji równa sie 0;**
- **przeniesienie – ustawiane, gdy wynikiem operacji jest przeniesienie. Używane przy operacjach arytmetycznych obejmujących wiele słów;**
- **równość - ustawiane, gdy wynikiem porównania logicznego jest równość;**
- **przepełnienie – wskazuje przepełnienie arytmetyczne;**
- **zezwolenie/blokowanie przerwania;**
- **nadzorca – wskazuje, czy procesor pracuje w trybie nadzorcy, czy w trybie użytkownika;**

Projektowanie listy rozkazów

Problemy projektowania:

- **repertuar operacji – ilość i złożoność dostępnych operacji;**
- **rodzaje danych – różne rodzaje danych, na których wykonywane są operacje;**
- **format rozkazu – długość rozkazu w bitach;**
- **rejestry – liczba rejestrów, do których mogą odnosić się rozkazy;**
- **Adresowanie –tryb/tryby specyfikacji adresów argumentów;**

Repertuar operacji:

Transfer danych:

- **Move – przeniesienie słowa lub bloku ze źródła do miejsca przeznaczenia;**
- **Store – przeniesienie słowa z procesora do pamięci;**
- **Load - przeniesienie słowa z pamięci do procesora;**
- **Exchange - zmiana zawartości źródła i miejsca przeznaczenia;**
- **Clear - przeniesienie słowa złożonego z zer do miejsca przeznaczenia;**
- **Set - przeniesienie słowa złożonego z 1 do miejsca przeznaczenia;**
- **Push-przeniesienie słowa ze źródła na wierzchołek stosu;**
- **Pop - przeniesienie słowa z wierzchołka stosu do wejścia przeznaczenia;**

Arytmetyczne

- **Add-obliczanie sumy dwóch argumentów;**
- **Substract-odejmowanie;**
- **Multiply-iloczyn;**
- **Divide-dzielenie;**
- **Absolute-zamiana argumentu na jego wartość bezwzględną;**
- **Negate-zmiana znaku argumentu;**
- **Increment-dodanie 1 do argumentu;**
- **Decrement-odjęcie 1 od argumentu;**

Logiczne

- **AND, OR, NOT, XOR - operacje na poziomie bitowym;**
- **Test - zbadanie określonego warunku;**
- **Compare - logiczne lub arytmetyczne porównanie dwóch argumentów ,na podstawie wyniku ustawienie znacznika(ów) stanu;**
- **Set control variable - klasa rozkazów ustalających elementy sterowania;**
- **Shift - przesunięcie bitów w lewo lub w prawo z prowadzeniem stałej na końcu;**
- **Rotate - przesunięcie bitów z łączeniem końca z początkiem;**

Przeniesienie sterowania

- **Jump – przeniesienie bezwarunkowe, ładuje określony adres do licznika programu;**
- **Jump Conditional – zbadanie określonego warunku i załadowanie adresu zgodnie z warunkiem;**
- **Jump to Subroutine – umieszczenie informacji kontrolnej bieżącego programu w znanym miejscu, skok do określonego adresu;**
- **Return – zmiana zawartości licznika rozkazów i innych rejestrów na dane przechowywane w znanym miejscu;**
- **Execute – pobranie argumentu z określonego miejsca i wykonanie go jako rozkazu bez modyfikacji licznika rozkazów;**
- **Skip – inkrementowanie licznika rozkazów w celu pominięcia kolejnego rozkazu;**
- **Skip Conditional – zbadanie określonego warunku i pominięcie (lub nie) kolejnego rozkazu;**
- **Halt – zatrzymanie wykonania programu;**
- **Wait – oczekiwanie na spełnienie określonego warunku;**

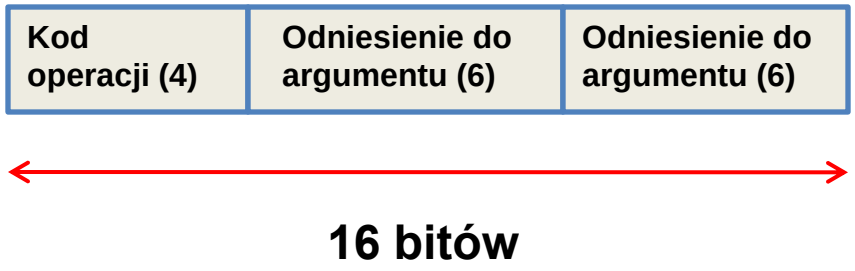
Wejście - wyjście

- **Input – przeniesienie danych, np. z pamięci do rejestru procesora;**
- **Output – przeniesienie danych z rejestru procesora do pamięci;**
- **Start I/O – rozkaz skierowany do modułu we-wy;**
- **Test I/O – przeniesienie informacji o stanie modułu we-wy.**

Elementy rozkazu maszynowego

- kod operacji jaka ma być wykonana;
- odniesienie do argumentów źródłowych;
- Odniesienie do wyniku;
- Odniesienie do następnego rozkazu;

Format rozkazu – rozkaz jest dzielony na pola odpowiadające elementom składowym rozkazu



Symboliczna reprezentacja kodów operacji, mnemoniki

- **ADD – dodaj**
- **SUB –odejmij**
- **MPY – pomnóż**
- **LOAD – ładuj z pamięci**
- **STOR – zapisz dane**

Przykład rozkazu:
ADD R, Y
Dodaj zawartość w pozycji danych Y do zawartości rejestru R,
Y – adres komórki pamięci

ADD R,Y - dodaj zawartość danych z komórki Y do zawartości rejestru R

Rozkazy maszynowe

Pisanie programów w języku maszynowym jest rzadkością . Programy pisze się w języku asemblerowym lub językach wysokiego poziomu

Przykład: X= X + Y

Załóżmy, że X i Y odpowiadają komórką pamięci o adresach 513, 514

W języku maszynowym mamy następujące rozkazy:

- Załaduj do rejestru zawartość komórki pamięci 513
- Dodaj zawartość komórki 514 do rejestru
- Przenieść zawartość rejestru do komórki 513

Liczba adresów w rozkazie

Jaka liczba adresów potrzebna jest w rozkazie:

- 2 adresy - odniesienie do argumentów;
- 1 adres - odniesienie do wyniku rozkazu
- 1 adres – odniesienie do następnego rozkazu

Jednak 4 adresowe rozkazy występują bardzo rzadko. Najczęściej np. adres następnego rozkazu jest domyślny, uzyskiwany z licznika rozkazów.

Przykład

$Y=(A-B)/(C+D*E)$

Rozkazy trójadresowe:

SUB Y, A, B	**Y ← A - B**
MPY T, D, E	**T ← D * E**
ADD T, T, C	**T ← T + C**
DIV Y, Y, T	**Y ← Y / T**

Przykład

Y=(A-B)/(C+D*E)

Rozkazy dwuadresowe:

MOVE Y, A	**Y ← A**
SUB Y, B	**Y ← Y - B**
MOVE T, D	**T ← D**
MPY T, E	**T ← T * E**
ADD T, C	**T ← T + C**
DIV Y, T	**Y ← Y / T**

Przykład

Y=(A-B)/(C+D*E)

Rozkazy jednoadresowe: **AC - akumulator**

LOAD D	AC ← D
MPY E	AC ← AC * E
ADD C	AC ← AC + C
STORE Y	Y ← AC
LOAD A	AC ← A
SUB B	AC ← AC - B
DIV Y	AC ← AC / Y
STORE Y	Y ← AC

Liczba adresów w rozkazie

- Większość współczesnych komputerów stosuje rozkazy 2 i 3 adresowe
- Jeśli procesor ma wiele rejestrów to operacje mogą być wykonywane na samych rejestrach
- Możliwe jest też by rozkazy funkcjonowały bez adresów. Takie rozkazy mogą być używane w przypadku organizacji pamięci zwanej stosem.