Architektura systemów komputerowych

Mariusz Wiśniewski

Politechnika Świętokrzyska w Kielcach
Katedra Informatyki

Arytmetyka maszyn cyfrowych

Plan wykładu

1. Typy danych w komputerach.
2. Układ arytmetyczno-logiczny.
3. Instrukcje zależne od ALU.
4. Superskalarność.

Cele

Wiedza na temat arytmetyki maszyn cyfrowych. Wiedza na temat operacji arytmetycznych, logicznych. Techniki projektowania mikroprocesora z uwzględnieniem specyfiki przetwarzanych danych.

Typy danych w komputerach

Typy danych w komputerach

podstawowe typy danych dostępne w komputerach

- liczby całkowite
- liczby niecałkowite
- ciągi znaków

Obliczenia w systemach komputerowych wykonuje się korzystając z cyfr binarnych o wartości 0 i 1, które służą do reprezentacji liczb.

Pojęcia:
- bit, bajt, słowo
- kod NKB i BCD
- kody U1, U2 i ZM

Typy danych w komputerach

podstawowe typy danych dostępne w komputerach

- liczby całkowite
- liczby niecałkowite
- ciągi znaków

Obliczenia w systemach komputerowych wykonuje się korzystając z cyfr binarnych o wartości 0 i 1, które służą do reprezentacji liczb.

Pojęcia:

- bit, bajt, słowo
- kod NKB i BCD
- kody U1, U2 i ZM

Bit jest najmniejszą jednostką logiczną, przenoszącą wartość, jaką może przyjąć cyfrowy układ logiczny.

Bajt jest najmniejszą jednostką informacji pamięci komputera, składający się z bitów – zazwyczaj z ośmiu: 1B = 8b.

Słowo maszynowe jest podstawową jednostką Informacji, jaką domyślnie przetwarza dany system komputerowy.

M.Wiśniewski

Typy danych w komputerach

podstawowe typy danych dostępne w komputerach

- liczby całkowite
- liczby niecałkowite
- ciągi znaków

Obliczenia w systemach komputerowych wykonuje się korzystając z cyfr binarnych o wartości 0 i 1, które służą do reprezentacji liczb.

Pojęcia:

- bit, bajt, słowo
- kod NKB i BCD
- kody U1, U2 i ZM

Bit jest najmniejszą jednostką logiczną, przenoszącą wartość, jaką może przyjąć cyfrowy układ logiczny ...

Naturalny kod binarny jest sposobem na reprezentację liczb dodatnich w dwójkowych systemach komputerowych. NKB jest pozycyjnym systemem liczbowym, gdzie podstawą jest cyfra 2. Postać liczby jest następująca:

$$b^n b^{n-1} b^{n-2} b \ldots b^1 b^0, \text{ gdzie } b = 2.$$

W systemach komputerowych stosuje się dwa sposoby zapisu liczb binarnych:

- najmłodszy bit znajduje się z prawej strony,
- najmłodszy bit znajduje się z lewej strony.

Kod BCD (Binary -Coded Decimal) służy do zapisu liczb w sposób polegający na zakodowaniu cyfr dziesiętnych liczby przez kolejne czwórki bitów liczby binarnej. Na przykład:

0 1 0 1	1 0 0 1	0 0 0 0	0 0 0 1
5	9	0	1

Typy danych w komputerach

podstawowe typy danych dostępne w komputerach

- liczby całkowite
- liczby niecałkowite
- ciągi znaków

Obliczenia w systemach komputerowych wykonuje się korzystając z cyfr binarnych o wartości 0 i 1, które służą do reprezentacji liczb.

Pojęcia:
- bit, bajt, słowo
- kod NKB i BCD
- kody U1, U2 i ZM

Bit jest najmniejszą jednostką logiczną, przenoszącą wartość, jaką może przyjąć cyfrowy układ logiczny ...

Naturalny kod binarny jest sposobem na reprezentację liczb dodatnich w dwójkowych systemach komputerowych. NKB jest pozycyjnym systemem liczbowym, gdzie podstawą jest cyfra 2. Postać liczby jest następująca:

$$b^n b^{n-1} b^{n-2} b \ldots b^1 b^0, \text{ gdzie } b = 2.$$

W systemach komputerowych stosuje się dwa sposoby zapisu liczb binarnych:
- najmłodszy bit znajduje się z prawej strony,
- najmłodszy bit znajduje się z lewej strony.

Kod BCD (Binary -Coded Decimal) służy do zapisu liczb w sposób polegający na zakodowaniu cyfr dziesiętnych liczby przez kolejne czwórki bitów liczby binarnej. Na przykład:

0 1 0 1	1 0 0 1	0 0 0 0	0 0 0 1
5	9	0	1

Kody uzupełnieniowe służą do zapisywania liczb ujemnych w systemach dwójkowych. W ogólności najstarszy bit liczby traktuje się jako znak liczby – 0 - liczba dodatnia, 1 - liczba ujemna. W praktyce stosuje się dwa kody liczbowe:
- kod U1 – kod uzupełnień do 1:
  Liczbę ujemną uzyskuje się przez negację bitów liczby zapisanej w NKB. W U1 występują dwa zera – dodatnie (000...00) i ujemne (111...11).
- kod U2 – kod uzupełnień do 2:
  Liczba ujemna powstaje przez zanegowanie liczby NKB i zwiększenie jej o 1. W kodzie występuje jedno zero. Z tego powodu kod jest powszechnie wykorzystywany we współczesnych systemach dwójkowych.

Innym sposobem na reprezentację liczby ujemnej jest postać znak-moduł, gdzie pierwszy bit liczby jest jej znakiem, a pozostałe bity są modułem liczby (wartością NKB).

Typy danych w komputerach

podstawowe typy danych dostępne w komputerach

- liczby całkowite
- liczby niecałkowite
- ciągi znaków

W systemach komputerowych stosuje się także liczby należące do zbioru liczb rzeczywistych.

Pojęcia:
- liczby zmiennoprzecinkowe
- liczby stałopozycyjne

Typy danych w komputerach

podstawowe typy danych dostępne w komputerach

- liczby całkowite
- liczby niecałkowite
- ciągi znaków

W systemach komputerowych stosuje się także liczby należące do zbioru liczb rzeczywistych.

Pojęcia:
- liczby zmiennoprzecinkowe
- liczby stałopozycyjne

W ogólności liczby zmiennoprzecinkowe mają postać:

$$L = S \cdot M \cdot B^E$$

gdzie:
- S - znak liczby, S = 1 - liczba dodatnia,
- M - znormalizowana mantysa liczby, część ułamkowa liczby,
- B - podstawa liczby, B = 2 dla systemów komputerowych,
- E - wykładnik liczby.

Normalizacja liczby polega na doprowadzenia mantysy liczby do postaci, w której najstarszy bit będzie równy 1. Podczas normalizacji modyfikuje się wykładnik liczby. W zapisie liczby zmiennoprzecinkowej pomija się pierwszy bit mantysy.

Liczby zmiennoprzecinkowe zostały zestandardyzowane w normie IEEE-754. Obecnie niemal we wszystkich systemach komputerowych stosuje się zapis liczb zgodny w powyższą normą.

Typy danych w komputerach

podstawowe typy danych dostępne w komputerach

- liczby całkowite
- liczby niecałkowite
- ciągi znaków

W systemach komputerowych stosuje się także liczby należące do zbioru liczb rzeczywistych.

Pojęcia:
- liczby zmiennoprzecinkowe
- liczby stałopozycyjne

Zapis stałopozycyjny liczb polega na umownym wskazaniu bitu liczby po którym znajdzie się przecinek oddzielający część całkowitą liczby od ułamkowej:

$$c_n c_{n-1} c_{n-2} \cdots c_1 c_0 . u_1 u_2 \cdots u_{m-1} u_m$$

gdzie: c_n – bity części całkowitej, u_m – bity liczby ułamkowej.

Ponieważ pozycja przecinka jest umowna – zazwyczaj decyduje o niej projektant systemu – to można dopasować zakres liczb do wymagań systemu. W tym wypadku bity liczby należy interpretować następująco:
- w części całkowitej – jako liczbę NKB,
- w części ułamkowej – każdy bit jest równy wartości dziesiętnej $h = 1 / 2^{(\text{liczba bitów ułamkowych})}$.

Przykłady:

```
  1011.0011011b = 11.2109375d        h = 0.0078125
+ 0001.0010011b =  1.1484375d
-----------------------------
= 1100.0101110b =  ?
```

W ogólności liczby zmiennoprzecinkowe mają postać:

$$L = S \cdot M \cdot B^E$$

gdzie:
- S - znak liczby, S = 1 - liczba dodatnia,
- M - znormalizowana mantysa liczby, część ułamkowa liczby,
- B - podstawa liczby, B = 2 dla systemów komputerowych,
- E - wykładnik liczby.

Normalizacja liczby polega na doprowadzenia mantysy liczby do postaci, w której najstarszy bit będzie równy 1. Podczas normalizacji modyfikuje się wykładnik liczby. W zapisie liczby zmiennoprzecinkowej pomija się pierwszy bit mantysy.

Liczby zmiennoprzecinkowe zostały zestandardyzowane w normie IEEE-754. Obecnie niemal we wszystkich systemach komputerowych stosuje się zapis liczb zgodny w powyższą normą.

Typy danych w komputerach

podstawowe typy danych dostępne w komputerach

- liczby całkowite
- liczby niecałkowite
- ciągi znaków

Znak wyświetlany na ekranie komputera jest reprezentacją graficzną liczby zapisanej w pamięci komputera. Zbiory takich liczb są znane pod pojęciem ciągu znaków.

Pojęcia:
- znak i ciąg znaków
- kod ASCII
- unicode

Typy danych w komputerach

podstawowe typy danych dostępne w komputerach

- liczby całkowite
- liczby niecałkowite
- ciągi znaków

Znak wyświetlany na ekranie komputera jest reprezentacją graficzną liczby zapisanej w pamięci komputera. Zbiory takich liczb są znane pod pojęciem ciągu znaków.

Pojęcia:
- znak i ciąg znaków
- kod ASCII
- unicode

W systemie komputerowym znak jest graficzną reprezentacją liczby (bajtu lub słowa) – jest kwestia umowną jak dany znak ma wygląd.

Zbiór bajtów zajmujących pewien obszar RAM może stanowić ciąg znaków. W zależności od umownej interpretacji przyjęło się, że ciąg znaków składa się:
- ze słowa (bajtu) określającego liczbę znaków w ciągu oraz bajtów (słów) odpowiadających znakom ciągu,
- z bajtów należących do ciągu znaków, przy czym ostatni bajt ciągu jest równy wartości liczbowej 0.

Odnośnie ciągów znakowych wykonuje się:
- łączenie dwóch ciągów,
- usuwanie znaków z ciągu,
- wyodrębnianie podciągu z ciągu.

Powyższe czynności polegają na wykonaniu rozkazów CPU służących do przenoszenia bajtów z jednego obszaru pamięci do drugiego.

Typy danych w komputerach

podstawowe typy danych dostępne w komputerach

- liczby całkowite
- liczby niecałkowite
- ciągi znaków

Znak wyświetlany na ekranie komputera jest reprezentacją graficzną liczby zapisanej w pamięci komputera. Zbiory takich liczb są znane pod pojęciem ciągu znaków.

Pojęcia:
- znak i ciąg znaków
- kod ASCII
- unicode

W systemie komputerowym znak jest graficzną reprezentacją liczby (bajtu lub słowa) – jest kwestia umowną jak dany znak ma wygląd.

Zbiór bajtów zajmujących pewien obszar RAM może stanowić ciąg znaków. W zależności od umownej interpretacji przyjęło się, że ciąg znaków składa się:
- ze słowa (bajtu) określającego liczbę znaków w ciągu oraz bajtów (słów) odpowiadających znakom ciągu,
- z bajtów należących do ciągu znaków, przy czym ostatni bajt ciągu jest równy wartości liczbowej 0.

Odnośnie ciągów znakowych wykonuje się:
- łączenie dwóch ciągów,
- usuwanie znaków z ciągu,
- wyodrębnianie podciągu z ciągu.

Powyższe czynności polegają na wykonaniu rozkazów CPU służących do przenoszenia bajtów z jednego obszaru pamięci do drugiego.

Kod ASCII jest standardem definiującym wygląd znaków w zakresie liczbowym od 0 do 127 (7-bitów). Znaki zostały podzielone na zakresy:
- kody sterujące terminalem lub drukarką (0 – 31 i 127),
- znaki drukowalne: białe, interpunkcyjne, cyfry, litery małe i duże (pozostałe 95 liczb).

Obecnie stosuje się kod ASCII 8-bitowy, w którym zostały zdefiniowano inne znaki, często zamieniane na znaki charakterystyczne dla danego języka (narodowe).

Typy danych w komputerach

podstawowe typy danych dostępne w komputerach

- liczby całkowite
- liczby niecałkowite
- ciągi znaków

Znak wyświetlany na ekranie komputera jest reprezentacją graficzną liczby zapisanej w pamięci komputera. Zbiory takich liczb są znane pod pojęciem ciągu znaków.

Pojęcia:
- znak i ciąg znaków
- kod ASCII
- unicode

Unicode jest standardem znaków w systemach komputerowych, obejmujący wszystkie znaki używane na świecie.

Najpowszechniej obecnie używaną odmianą unicode jest system UTF-8, gdzie jeden znak zapisuje się za pomocą 1, 2 lub 3 bajtów, przy czym wartość 0 nie jest przypisana do żadnego znaku (pozwala na tworzenie ciągów ASCIIZ).

Kod ASCII jest standardem definiującym wygląd znaków w zakresie liczbowym od 0 do 127 (7-bitów). Znaki zostały podzielone na zakresy:
- kody sterujące terminalem lub drukarką (0 – 31 i 127),
- znaki drukowalne: białe, interpunkcyjne, cyfry, litery małe i duże (pozostałe 95 liczb).

Obecnie stosuje się kod ASCII 8-bitowy, w którym zostały zdefiniowano inne znaki, często zamieniane na znaki charakterystyczne dla danego języka (narodowe).

W systemie komputerowym znak jest graficzną reprezentacją liczby (bajtu lub słowa) – jest kwestia umowną jak dany znak ma wygląd.

Zbiór bajtów zajmujących pewien obszar RAM może stanowić ciąg znaków. W zależności od umownej interpretacji przyjęło się, że ciąg znaków składa się:
- ze słowa (bajtu) określającego liczbę znaków w ciągu oraz bajtów (słów) odpowiadających znakom ciągu,
- z bajtów należących do ciągu znaków, przy czym ostatni bajt ciągu jest równy wartości liczbowej 0.

Odnośnie ciągów znakowych wykonuje się:
- łączenie dwóch ciągów,
- usuwanie znaków z ciągu,
- wyodrębnianie podciągu z ciągu.

Powyższe czynności polegają na wykonaniu rozkazów CPU służących do przenoszenia bajtów z jednego obszaru pamięci do drugiego.

Układ arytmetyczno-logiczny

Układ arytmetyczno-logiczny

budowa i działanie układu ALU procesora

- ALU
- algorytm sumowania
- algorytm mnożenia
- algorytm dzielenia

Zasadniczo komputery wykonują dwie czynności: przesyłanie danych oraz obliczenia. W drugim przypadku komputer musi posiadać do tego zadania specjalnie opracowany moduł.

Pojęcia:
- rejestry
- bity statusu
- konstrukcja

Układ arytmetyczno-logiczny

budowa i działanie układu ALU procesora

- ALU
- algorytm sumowania
- algorytm mnożenia
- algorytm dzielenia

Zasadniczo komputery wykonują dwie czynności: przesyłanie danych oraz obliczenia. W drugim przypadku komputer musi posiadać do tego zadania specjalnie opracowany moduł.

Pojęcia:

- rejestry
- bity statusu
- konstrukcja

Układ ALU posiada pewną liczbę rejestrów dostępnych tylko dla niego, w których są np. przechowywane wyniki częściowe obliczeń. W minimalnej implementacji ALU udostępnia dwa rejestry:

- rejestr akumulatora – będący argumentem rozkazów oraz miejscem na wyniki,
- rejestr bitów statusu.

Dostępność tych rejestrów dla programisty jest zależna od modelu programowania CPU.

Układ arytmetyczno-logiczny

budowa i działanie układu ALU procesora

- ALU
- algorytm sumowania
- algorytm mnożenia
- algorytm dzielenia

Zasadniczo komputery wykonują dwie czynności: przesyłanie danych oraz obliczenia. W drugim przypadku komputer musi posiadać do tego zadania specjalnie opracowany moduł.

Pojęcia:

- rejestry
- bity statusu
- konstrukcja

Układ ALU posiada pewną liczbę rejestrów dostępnych tylko dla niego, w których są np. przechowywane wyniki częściowe obliczeń. W minimalnej implementacji ALU udostępnia dwa rejestry:

- rejestr akumulatora – będący argumentem rozkazów oraz miejscem na wyniki,
- rejestr bitów statusu.

Dostępność tych rejestrów dla programisty jest zależna od modelu programowania CPU.

Zazwyczaj w trakcie wykonywania obliczeń układ ALU generuje dodatkowe informacje charakteryzujące wynik, nazywane bitami statusu:

- cf - znacznik przeniesienia,
- pf - znacznik parzystości,
- zf - znacznik zera,
- sf - znacznik znaku,
- of - znacznik nadmiaru w kodzie U2, obliczany wg zależności $of = cf_n \text{ xor } cf_{n-1}$.

Układ arytmetyczno-logiczny

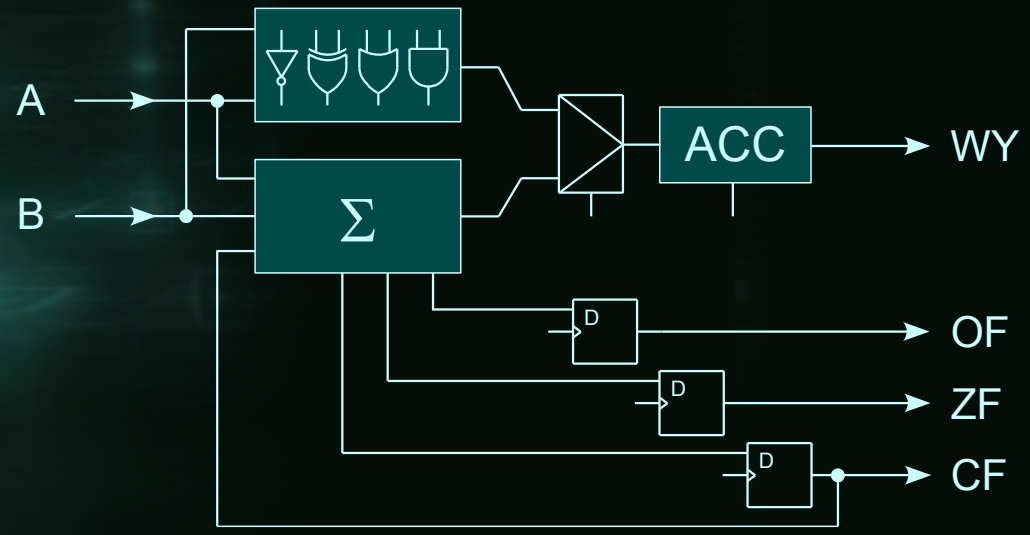
budowa i działanie układu ALU procesora

- ALU
  - algorytm sumowania
  - algorytm mnożenia
  - algorytm dzielenia

Zasadniczo komputery wykonują dwie czynności: przesyłanie danych oraz obliczenia. W drugim przypadku komputer musi posiadać do tego zadania specjalnie opracowany moduł.

Pojęcia:
- rejestry
- bity statusu
- konstrukcja

W układzie ALU mogą znajdować się: moduły logiczne, sumatory, mnożarki i dzielarki, ewentualnie logika sterująca, jeśli operacje ALU nie są mikrooperacjami.



Układ ALU posiada pewną liczbę rejestrów dostępnych tylko dla niego, w których są np. przechowywane wyniki częściowe obliczeń. W minimalnej implementacji ALU udostępnia dwa rejestry:
- rejestr akumulatora – będący argumentem rozkazów oraz miejscem na wyniki,
- rejestr bitów statusu.

Dostępność tych rejestrów dla programisty jest zależna od modelu programowania CPU.

Zazwyczaj w trakcie wykonywania obliczeń układ ALU generuje dodatkowe informacje charakteryzujące wynik, nazywane bitami statusu:
- cf - znacznik przeniesienia,
- pf - znacznik parzystości,
- zf - znacznik zera,
- sf - znacznik znaku,
- of - znacznik nadmiaru w kodzie U2, obliczany wg zależności of = cf_n xor cf_{n-1} .

Układ arytmetyczno-logiczny

budowa i działanie układu ALU procesora

- ALU
- algorytm sumowania
- algorytm mnożenia
- algorytm dzielenia

Sumowanie jest podstawową operacją wykonuwaną przez ALU. Zależnie od kodu liczbowego proces sumowania przebiega nieco odmiennie.

Operacja:
- dodawanie w NKB
- dodawanie w U1
- dodawanie i odejmowanie w kodzie U2

Układ arytmetyczno-logiczny

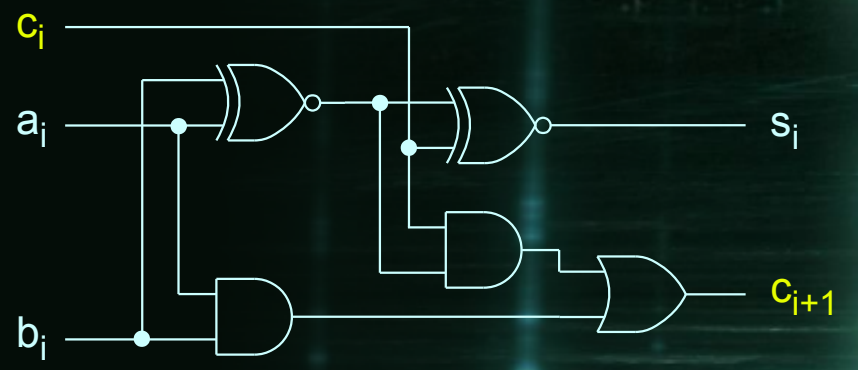
budowa i działanie układu ALU procesora

- ALU
- algorytm sumowania
- algorytm mnożenia
- algorytm dzielenia

Sumowanie jest podstawową operacją wykonuwaną przez ALU. Zależnie od kodu liczbowego proces sumowania przebiega nieco odmiennie.

Operacja:
- dodawanie w NKB
- dodawanie w U1
- dodawanie i odejmowanie w kodzie U2

Sumowanie odbywa się przez zastosowanie do każdych dwóch bitów operandów funkcji sumatora pełnego:



Układ arytmetyczno-logiczny

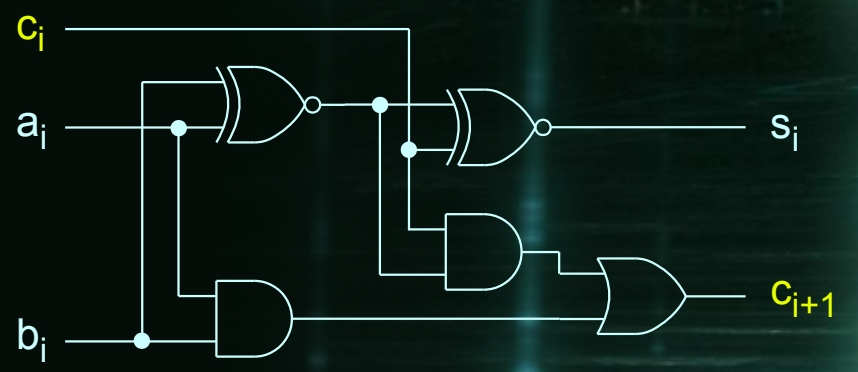
budowa i działanie układu ALU procesora

- ALU
- algorytm sumowania
- algorytm mnożenia
- algorytm dzielenia

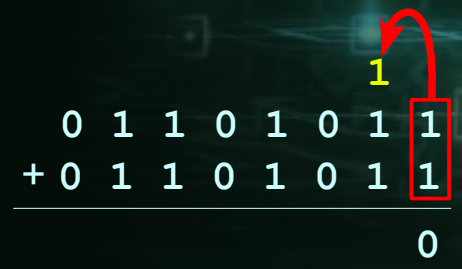
Sumowanie jest podstawową operacją wykonuwaną przez ALU. Zależnie od kodu liczbowego proces sumowania przebiega nieco odmiennie.

Operacja:
- dodawanie w NKB
- dodawanie w U1
- dodawanie i odejmowanie w kodzie U2

Sumowanie odbywa się przez zastosowanie do każdych dwóch bitów operandów funkcji sumatora pełnego:



Przykład:



Układ arytmetyczno-logiczny

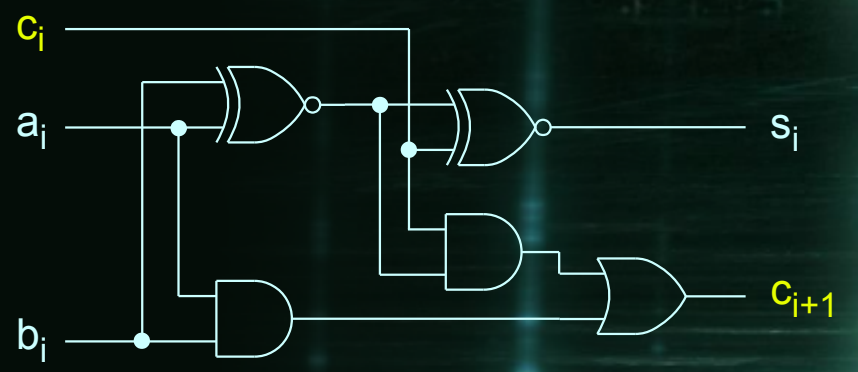
budowa i działanie układu ALU procesora

- ALU
- algorytm sumowania
- algorytm mnożenia
- algorytm dzielenia

Sumowanie jest podstawową operacją wykonuwaną przez ALU. Zależnie od kodu liczbowego proces sumowania przebiega nieco odmiennie.

Operacja:
- dodawanie w NKB
- dodawanie w U1
- dodawanie i odejmowanie w kodzie U2

Sumowanie odbywa się przez zastosowanie do każdych dwóch bitów operandów funkcji sumatora pełnego:



Przykład:

```
  1 1   1   1 1
  0 1 1 0 1 0 1 1
+ 0 1 1 0 1 0 1 1
-----------------
  1 1 0 1 0 1 1 0
```

Układ arytmetyczno-logiczny

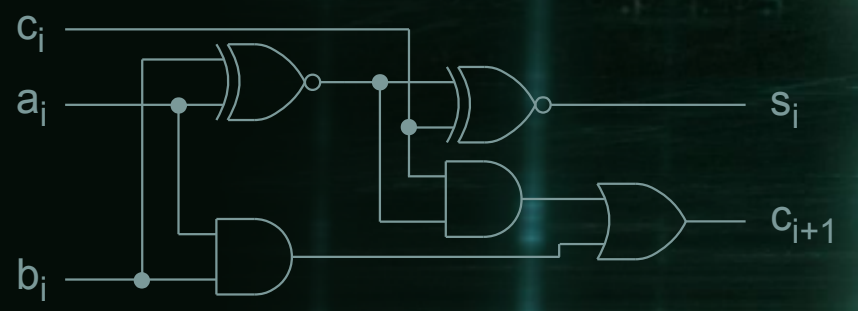
budowa i działanie układu ALU procesora

- ALU
- algorytm sumowania
- algorytm mnożenia
- algorytm dzielenia

Sumowanie jest podstawową operacją wykonuwaną przez ALU. Zależnie od kodu liczbowego proces sumowania przebiega nieco odmiennie.

Operacja:
- dodawanie w NKB
- dodawanie w U1
- dodawanie i odejmowanie w kodzie U2

Sumowanie odbywa się przez zastosowanie do każdych dwóch bitów operandów funkcji sumatora pełnego:



Liczby w kodzie U1 dodaje się bit po bicie, łącznie z bitem znaku. Jeśli pojawi się przeniesienie, należy je dodać do wyniku.

Przykład: obie liczby ujemne.

-13:	1	1	0	0	1	0
-11:	1	1	0	1	0	0
	(1)	1	0	0	1	1	0
							1
-24:	1	0	0	1	1	1

Układ arytmetyczno-logiczny

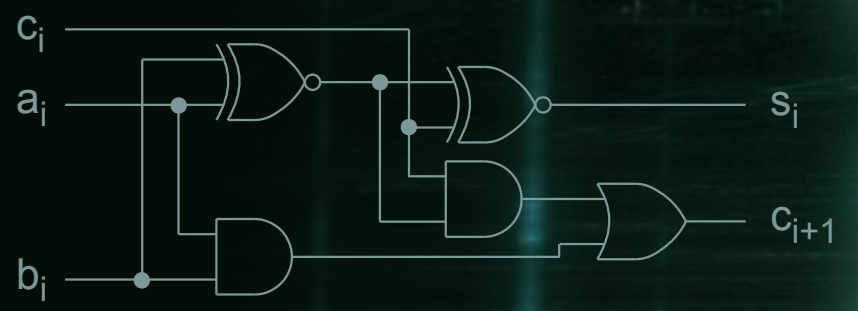
budowa i działanie układu ALU procesora

- ALU
- algorytm sumowania
- algorytm mnożenia
- algorytm dzielenia

Sumowanie jest podstawową operacją wykonuwaną przez ALU. Zależnie od kodu liczbowego proces sumowania przebiega nieco odmiennie.

Operacja:
- dodawanie w NKB
- dodawanie w U1
- dodawanie i odejmowanie w kodzie U2

Sumowanie odbywa się przez zastosowanie do każdych dwóch bitów operandów funkcji sumatora pełnego:



Liczby w kodzie U1 dodaje się bit po bicie, łącznie z bitem znaku. Jeśli pojawi się przeniesienie, należy je dodać do wyniku.

Przykład: liczby o przeciwnych znakach (1).

```
+13:  0 0 1 1 0 1
-11:  1 1 0 1 0 0
      ---------
      0 0 0 0 0 1
              1
      ---------
 +2:  0 0 0 0 1 0
```

Układ arytmetyczno-logiczny

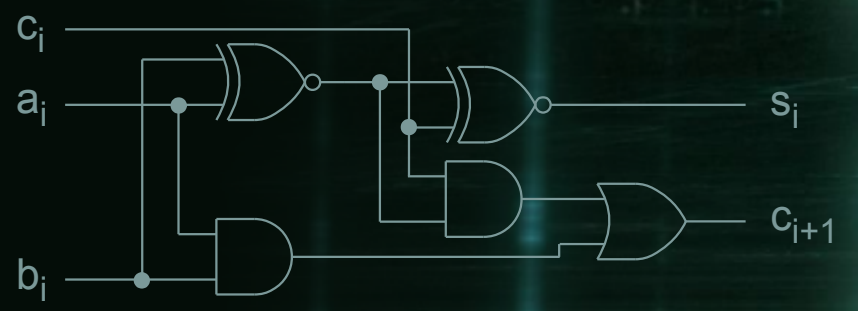
budowa i działanie układu ALU procesora

- ALU
- algorytm sumowania
- algorytm mnożenia
- algorytm dzielenia

Sumowanie jest podstawową operacją wykonuwaną przez ALU. Zależnie od kodu liczbowego proces sumowania przebiega nieco odmiennie.

Operacja:
- dodawanie w NKB
- dodawanie w U1
- dodawanie i odejmowanie w kodzie U2

Sumowanie odbywa się przez zastosowanie do każdych dwóch bitów operandów funkcji sumatora pełnego:



Liczby w kodzie U1 dodaje się bit po bicie, łącznie z bitem znaku. Jeśli pojawi się przeniesienie, należy je dodać do wyniku.

Przykład: liczby o przeciwnych znakach (2).

-11:	1	1	0	1	0	0
+13:	0	0	1	1	0	1
	0	0	0	0	0	1
						1
+2:	0	0	0	0	1	0

Układ arytmetyczno-logiczny

budowa i działanie układu ALU procesora

- ALU
- algorytm sumowania
- algorytm mnożenia
- algorytm dzielenia

Sumowanie jest podstawową operacją wykonuwaną przez ALU. Zależnie od kodu liczbowego proces sumowania przebiega nieco odmiennie.

Operacja:
- dodawanie w NKB
- dodawanie w U1
- dodawanie i odejmowanie w kodzie U2

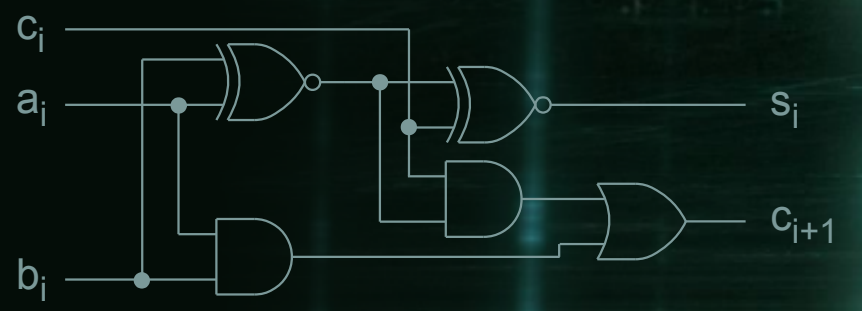
Liczby w kodzie U2 dodaje się bit po bicie, łącznie z bitem znaku, przy czym należy ignorować pojawiające się przeniesienie.

Przykłady:

-13:	1	1	0	0	1	1		-11:	1	1	0	0	1	1
-11:	1	1	0	1	0	1		+13:	0	0	1	1	0	1
-24:	1	0	1	0	0	0		-2:	1	1	1	1	1	0

-11:	1	1	0	1	0	1		+11:	0	0	1	0	1	1
+13:	0	0	1	1	0	1		-13:	1	1	0	0	1	1
+2:	0	0	0	0	1	0		-2:	1	1	1	1	1	0

Sumowanie odbywa się przez zastosowanie do każdych dwóch bitów operandów funkcji sumatora pełnego:



Liczby w kodzie U1 dodaje się bit po bicie, łącznie z bitem znaku. Jeśli pojawi się przeniesienie, należy je dodać do wyniku.

Przykład: liczby o przeciwnych znakach (2).

-11:	1	1	0	1	0	0	
+13:	0	0	1	1	0	1	
	0	0	0	0	0	1	
						1	
+2:	0	0	0	0	1	0	

Układ arytmetyczno-logiczny

budowa i działanie układu ALU procesora

- ALU
- algorytm sumowania
- algorytm mnożenia
- algorytm dzielenia

Algorytm mnożenia liczb binarnych jest złożony i zazwyczaj do skompletowania wyniku wymaga wielu cykli.

Mnożenie:
- w kodzie NKB
- w kodach U1 i U2

Układ arytmetyczno-logiczny

budowa i działanie układu ALU procesora

- ALU
- algorytm sumowania
- algorytm mnożenia
- algorytm dzielenia

Algorytm mnożenia liczb binarnych jest złożony i zazwyczaj do skompletowania wyniku wymaga wielu cykli.

Mnożenie:
- w kodzie NKB
- w kodach U1 i U2

W ogólności mnożenie polega na cyklicznym przeglądaniu mnożnika i przesuwaniu mnożnej o jeden bit w lewo lub w prawo oraz sumowaniu wyniku częściowego.

Przykład:

```
    0 1 0 0 1 0
x   1 0 0 1 0 1
```

Układ arytmetyczno-logiczny

budowa i działanie układu ALU procesora

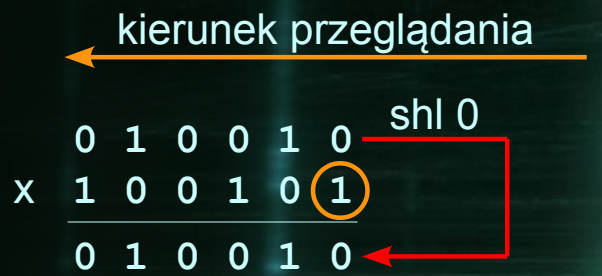
- ALU
- algorytm sumowania
- algorytm mnożenia
- algorytm dzielenia

Algorytm mnożenia liczb binarnych jest złożony i zazwyczaj do skompletowania wyniku wymaga wielu cykli.

Mnożenie:
- w kodzie NKB
- w kodach U1 i U2

W ogólności mnożenie polega na cyklicznym przeglądaniu mnożnika i przesuwaniu mnożnej o jeden bit w lewo lub w prawo oraz sumowaniu wyniku częściowego.

Przykład:



Układ arytmetyczno-logiczny

budowa i działanie układu ALU procesora

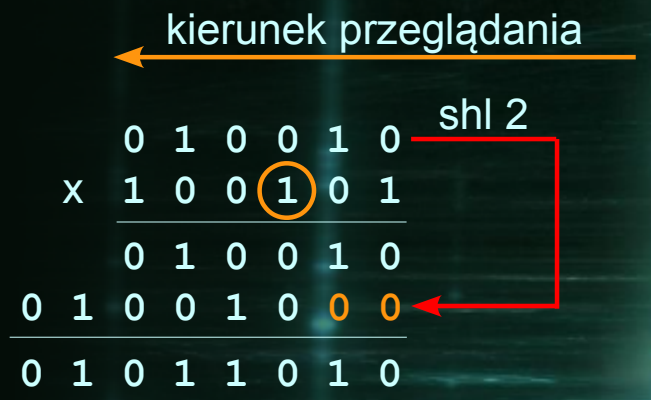
- ALU
- algorytm sumowania
- algorytm mnożenia
- algorytm dzielenia

Algorytm mnożenia liczb binarnych jest złożony i zazwyczaj do skompletowania wyniku wymaga wielu cykli.

Mnożenie:
- w kodzie NKB
- w kodach U1 i U2

W ogólności mnożenie polega na cyklicznym przeglądaniu mnożnika i przesuwaniu mnożnej o jeden bit w lewo lub w prawo oraz sumowaniu wyniku częściowego.

Przykład:



Układ arytmetyczno-logiczny

budowa i działanie układu ALU procesora

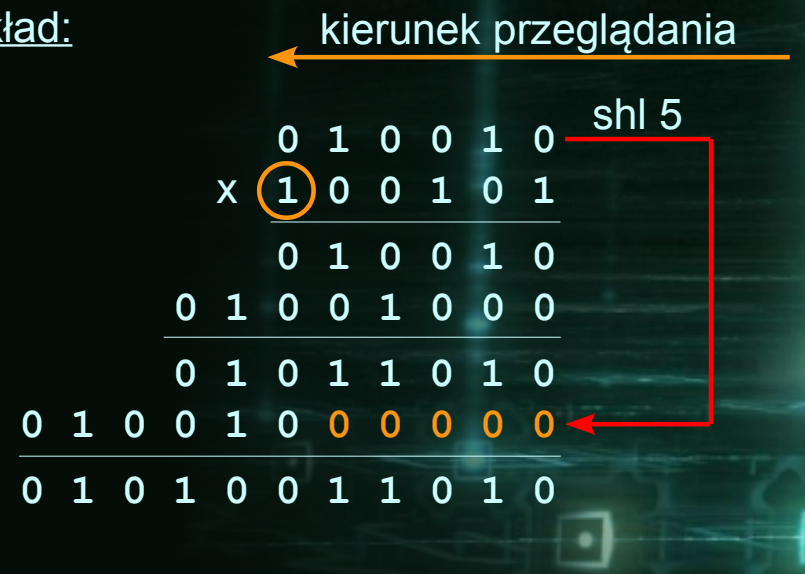
- ALU
- algorytm sumowania
- algorytm mnożenia
- algorytm dzielenia

Algorytm mnożenia liczb binarnych jest złożony i zazwyczaj do skompletowania wyniku wymaga wielu cykli.

Mnożenie:
- w kodzie NKB
- w kodach U1 i U2

W ogólności mnożenie polega na cyklicznym przeglądaniu mnożnika i przesuwaniu mnożnej o jeden bit w lewo lub w prawo oraz sumowaniu wyniku częściowego.

Przykład:



Układ arytmetyczno-logiczny

budowa i działanie układu ALU procesora

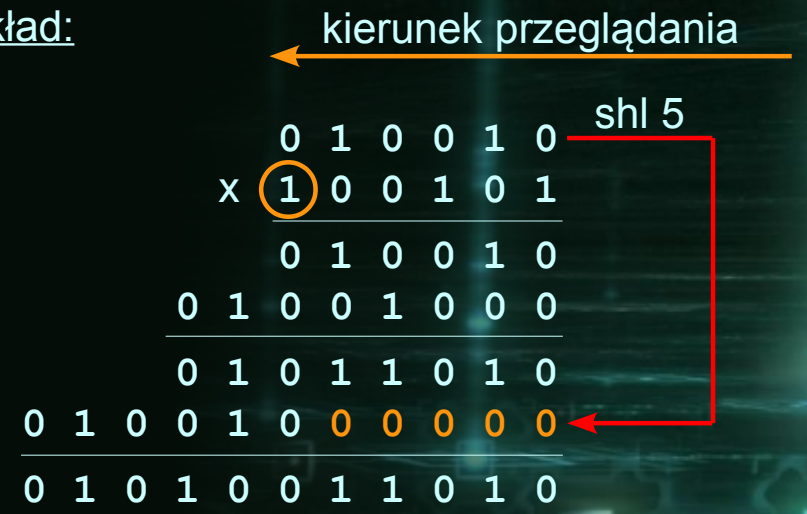
- ALU
- algorytm sumowania
- algorytm mnożenia
- algorytm dzielenia

Algorytm mnożenia liczb binarnych jest złożony i zazwyczaj do skompletowania wyniku wymaga wielu cykli.

Mnożenie:
- w kodzie NKB
- w kodach U1 i U2

W ogólności mnożenie polega na cyklicznym przeglądaniu mnożnika i przesuwaniu mnożnej o jeden bit w lewo lub w prawo oraz sumowaniu wyniku częściowego.

Przykład:



Implementacja:
- zazwyczaj mnożnik przegląda się od prawej do lewej, a mnożną przesuwa się bit po bicie w trakcie przeglądania mnożnika,
- wynik wymaga rejestru o liczbie bitów równej sumie bitów operandów.

Układ arytmetyczno-logiczny

budowa i działanie układu ALU procesora

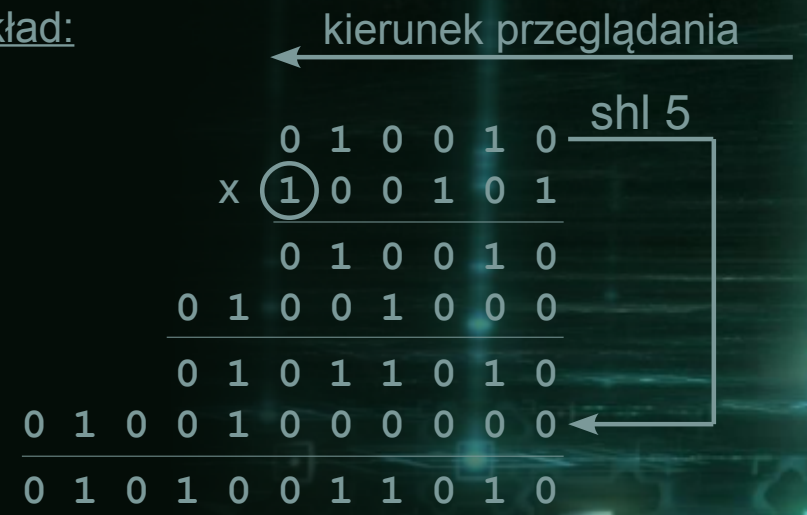
- ALU
- algorytm sumowania
- algorytm mnożenia
- algorytm dzielenia

Algorytm mnożenia liczb binarnych jest złożony i zazwyczaj do skompletowania wyniku wymaga wielu cykli.

Mnożenie:
- w kodzie NKB
- w kodach U1 i U2

W ogólności mnożenie polega na cyklicznym przeglądaniu mnożnika i przesuwaniu mnożnej o jeden bit w lewo lub w prawo oraz sumowaniu wyniku częściowego.

Przykład:



Mnożenie w kodach U1 i U2 wykonuje się dwojako:
- bezpośrednio mnożąc liczby przez siebie i następnie wykonując korekcję wyniku,
- metodami dedykowanymi dla kodu i znaków operandów.

Algorytm dla kodu U2: $A \cdot B$, gdzie $A = -a$, $B = +b$
1. każda jedynka w mnożniku wymaga dodania do wyniku częściowego przesuniętej w lewo mnożnej,
2. w U2 po każdym przesunięciu mnożnej na jej najmniej znaczące pozycje wpisuje się wartość 0,
3. po przejrzeniu wszystkich bitów mnożnika wynik jest poprawny.

Implementacja:
- zazwyczaj mnożnik przegląda się od prawej do lewej, a mnożną przesuwa się bit po bicie w trakcie przeglądania mnożnika,
- wynik wymaga rejestru o liczbie bitów równej sumie bitów operandów.

Układ arytmetyczno-logiczny

budowa i działanie układu ALU procesora

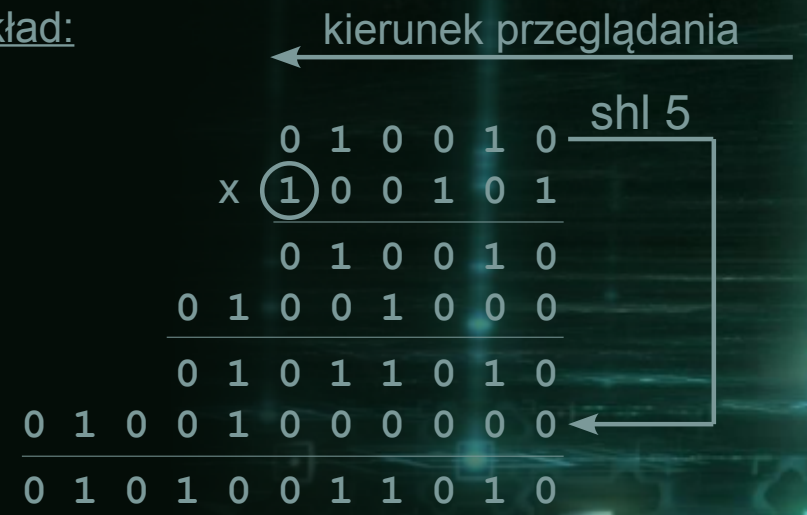
- ALU
- algorytm sumowania
- algorytm mnożenia
- algorytm dzielenia

Algorytm mnożenia liczb binarnych jest złożony i zazwyczaj do skompletowania wyniku wymaga wielu cykli.

Mnożenie:
- w kodzie NKB
- w kodach U1 i U2

W ogólności mnożenie polega na cyklicznym przeglądaniu mnożnika i przesuwaniu mnożnej o jeden bit w lewo lub w prawo oraz sumowaniu wyniku częściowego.

Przykład:



Mnożenie w kodach U1 i U2 wykonuje się dwojako:
- bezpośrednio mnożąc liczby przez siebie i następnie wykonując korekcję wyniku,
- metodami dedykowanymi dla kodu i znaków operandów.

Algorytm dla kodu U2: $A \cdot B$, gdzie $A = +a/-a$, $B = -b$

1. mnożnik należy uzupełnić do 2,
2. mnożną należy uzupełnić do 1,
3. bity z prawej strony mnożnej i mnożnika powinny mieć wartości wynikające z zastosowanego przekształcenia,
4. dla każdej kolejnej jedynki w uzupełnionym do 2 mnożniku do iloczynu częściowego dodaje się odpowiednio przesuniętą w lewo, uzupełnioną do 1 mnożną,
5. do iloczynu częściowego dodaje się 1, każdorazowo przesunięte w lewo (tak jak mnożna), uzupełniane na najmniej znaczących pozycjach wartością 0.

Implementacja:
- zazwyczaj mnożnik przegląda się od prawej do lewej, a mnożną przesuwa się bit po bicie w trakcie przeglądania mnożnika,
- wynik wymaga rejestru o liczbie bitów równej sumie bitów operandów.

Układ arytmetyczno-logiczny

budowa i działanie układu ALU procesora

- ALU
- algorytm sumowania
- algorytm mnożenia
- algorytm dzielenia

Algorytm mnożenia liczb binarnych jest złożony i zazwyczaj do skompletowania wyniku wymaga wielu cykli.

Mnożenie:
- w kodzie NKB
- w kodach U1 i U2

W ogólności mnożenie polega na cyklicznym przeglądaniu mnożnika i przesuwaniu mnożnej o jeden bit w lewo lub w prawo oraz sumowaniu wyniku częściowego.

Przykład:

kierunek przeglądania

shl 5

```
      0 1 0 0 1 0
    x (1) 0 0 1 0 1
      ---------------
      0 1 0 0 1 0
    0 1 0 0 1 0 0 0
    ---------------
    0 1 0 1 1 0 1 0
0 1 0 0 1 0 0 0 0 0 0
---------------------
0 1 0 1 0 0 1 1 0 1 0
```

Mnożenie w kodach U1 i U2 wykonuje się dwojako:
- bezpośrednio mnożąc liczby przez siebie i następnie wykonując korekcję wyniku,
- metodami dedykowanymi dla kodu i znaków operandów.

Przykład dla kodu U2: A · B, gdzie A = 13, B = -11

```
          1 1 0 0 1 0  <-- (001101)U2
        x 0 0 1 0 1 1  <-- (110101)U2
    ------------------
  1 1 1 1 1 1 1 1 0 0 1 0
                    1
    ------------------
  1...1 1 0 0 1 1
```

„jedynka“ z pkt. 3 algorytmu.

iloczyn częściowy

Implementacja:
- zazwyczaj mnożnik przegląda się od prawej do lewej, a mnożną przesuwa się bit po bicie w trakcie przeglądania mnożnika,
- wynik wymaga rejestru o liczbie bitów równej sumie bitów operandów.

Układ arytmetyczno-logiczny

budowa i działanie układu ALU procesora

- ALU
- algorytm sumowania
- algorytm mnożenia
- algorytm dzielenia

Algorytm mnożenia liczb binarnych jest złożony i zazwyczaj do skompletowania wyniku wymaga wielu cykli.

Mnożenie:
- w kodzie NKB
- w kodach U1 i U2

W ogólności mnożenie polega na cyklicznym przeglądaniu mnożnika i przesuwaniu mnożnej o jeden bit w lewo lub w prawo oraz sumowaniu wyniku częściowego.

Przykład:

kierunek przeglądania

shl 5

```
      0 1 0 0 1 0
    x (1) 0 0 1 0 1
      ---------------
      0 1 0 0 1 0
    0 1 0 0 1 0 0 0
    ---------------
    0 1 0 1 1 0 1 0
0 1 0 0 1 0 0 0 0 0
-------------------
0 1 0 1 0 0 1 1 0 1 0
```

Mnożenie w kodach U1 i U2 wykonuje się dwojako:
- bezpośrednio mnożąc liczby przez siebie i następnie wykonując korekcję wyniku,
- metodami dedykowanymi dla kodu i znaków operandów.

Przykład dla kodu U2: A · B, gdzie A = 13, B = -11

```
                1 1 0 0 1 0  <-- (001101)U2
            x   0 0 1 0 1 1  <-- (110101)U2
                             ...
  1 1 1 1 1 1 1 1 0 0 1 1
  1 1 1 1 1 1 1 1 0 0 1 0 0
                      1 0
  ---------------------------
  1...1 0 1 1 0 0 1
```

poprzedni iloczyn cz.

„jedynka“ z pkt. 3 algorytmu.

nowy iloczyn częściowy

Implementacja:
- zazwyczaj mnożnik przegląda się od prawej do lewej, a mnożną przesuwa się bit po bicie w trakcie przeglądania mnożnika,
- wynik wymaga rejestru o liczbie bitów równej sumie bitów operandów.

Układ arytmetyczno-logiczny

budowa i działanie układu ALU procesora

- ALU
- algorytm sumowania
- algorytm mnożenia
- algorytm dzielenia

Algorytm mnożenia liczb binarnych jest złożony i zazwyczaj do skompletowania wyniku wymaga wielu cykli.

Mnożenie:
- w kodzie NKB
- w kodach U1 i U2

W ogólności mnożenie polega na cyklicznym przeglądaniu mnożnika i przesuwaniu mnożnej o jeden bit w lewo lub w prawo oraz sumowaniu wyniku częściowego.

Przykład:

kierunek przeglądania

shl 5

```
      0 1 0 0 1 0
    x (1) 0 0 1 0 1
      ---------------
      0 1 0 0 1 0
    0 1 0 0 1 0 0 0
    ---------------
    0 1 0 1 1 0 1 0
0 1 0 0 1 0 0 0 0 0
---------------------
0 1 0 1 0 0 1 1 0 1 0
```

Mnożenie w kodach U1 i U2 wykonuje się dwojako:
- bezpośrednio mnożąc liczby przez siebie i następnie wykonując korekcję wyniku,
- metodami dedykowanymi dla kodu i znaków operandów.

Przykład dla kodu U2: A · B, gdzie A = 13, B = -11

```
                1 1 0 0 1 0  <-- (001101)U2
            x   0 0 1 0 1 1  <-- (110101)U2
                             ...
poprzedni   1 1 1 1 1 1 0 1 1 0 0 1
iloczyn cz. 1 1 1 1 1 0 0 1 0 0 0 0
                      1 0 0 0   <-- „jedynka“ z pkt. 3 algorytmu.
            -------------------
            1 1 1 1 0 1 1 1 0 0 0 1
            wynik
```

Implementacja:
- zazwyczaj mnożnik przegląda się od prawej do lewej, a mnożną przesuwa się bit po bicie w trakcie przeglądania mnożnika,
- wynik wymaga rejestru o liczbie bitów równej sumie bitów operandów.

Układ arytmetyczno-logiczny

budowa i działanie układu ALU procesora

- ALU
- algorytm sumowania
- algorytm mnożenia
- algorytm dzielenia

Dzielenie liczb binarnych również jest złożoną operacją. W typowym CPU operacja dzielenia zajmuje ponadprzeciętną liczbę cykli.

Dzielenie:
- w kodzie NKB
- w kodzie U2

Układ arytmetyczno-logiczny

budowa i działanie układu ALU procesora

- ALU
- algorytm sumowania
- algorytm mnożenia
- algorytm dzielenia

Dzielenie liczb binarnych również jest złożoną operacją. W typowym CPU operacja dzielenia zajmuje ponadprzeciętną liczbę cykli.

Dzielenie:
- w kodzie NKB
- w kodzie U2

Wynikiem dzielenia liczb binarnych są iloraz i reszta z dzielenia. Wyznaczenie bitów ilorazu wymaga:
- przesunięcia dzielnika i odjęcie (dodanie dopełnienia) od ilorazu częściowego, jeśli znak wyniku nie ulegnie zmianie – bit ilorazu równy 1,
- wykonanie samego przesunięcia, jeśli odejmowanie powoduje zmianę znaku – bit ilorazu równy 0.

Przykład:

```
                                ?
0 1 1 0 1 / 0 0 1 0 0 1 0 0 0 1
```

Układ arytmetyczno-logiczny

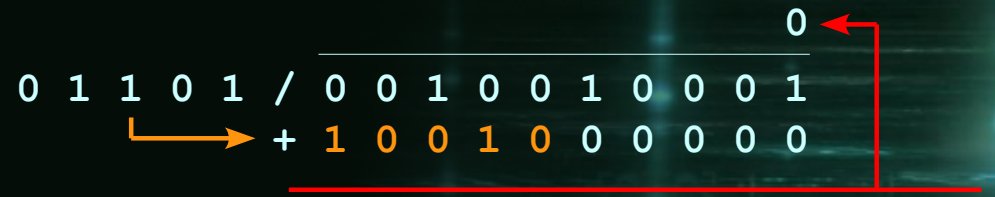
budowa i działanie układu ALU procesora

Dzielenie liczb binarnych również jest złożoną operacją. W typowym CPU operacja dzielenia zajmuje ponadprzeciętną liczbę cykli.

Dzielenie:
- w kodzie NKB
- w kodzie U2

Wynikiem dzielenia liczb binarnych są iloraz i reszta z dzielenia. Wyznaczenie bitów ilorazu wymaga:
- przesunięcia dzielnika i odjęcie (dodanie dopełnienia) od ilorazu częściowego, jeśli znak wyniku nie ulegnie zmianie – bit ilorazu równy 1,
- wykonanie samego przesunięcia, jeśli odejmowanie powoduje zmianę znaku – bit ilorazu równy 0.

Przykład:



Na początku dzielnik jest zazwyczaj większy od dzielnej, zatem pierwsze odejmowanie nie będzie dozwolone – do wyniku należy wpisać 0.

Układ arytmetyczno-logiczny

budowa i działanie układu ALU procesora

Dzielenie liczb binarnych również jest złożoną operacją. W typowym CPU operacja dzielenia zajmuje ponadprzeciętną liczbę cykli.

Dzielenie:
- w kodzie NKB
- w kodzie U2

Wynikiem dzielenia liczb binarnych są iloraz i reszta z dzielenia. Wyznaczenie bitów ilorazu wymaga:
- przesunięcia dzielnika i odjęcie (dodanie dopełnienia) od ilorazu częściowego, jeśli znak wyniku nie ulegnie zmianie – bit ilorazu równy 1,
- wykonanie samego przesunięcia, jeśli odejmowanie powoduje zmianę znaku – bit ilorazu równy 0.

Przykład:

Układ arytmetyczno-logiczny

budowa i działanie układu ALU procesora

- ALU
- algorytm sumowania
- algorytm mnożenia
- algorytm dzielenia

Dzielenie liczb binarnych również jest złożoną operacją. W typowym CPU operacja dzielenia zajmuje ponadprzeciętną liczbę cykli.

Dzielenie:
- w kodzie NKB
- w kodzie U2

Wynikiem dzielenia liczb binarnych są iloraz i reszta z dzielenia. Wyznaczenie bitów ilorazu wymaga:
- przesunięcia dzielnika i odjęcie (dodanie dopełnienia) od ilorazu częściowego, jeśli znak wyniku nie ulegnie zmianie – bit ilorazu równy 1,
- wykonanie samego przesunięcia, jeśli odejmowanie powoduje zmianę znaku – bit ilorazu równy 0.

Przykład:

Ze względu na dopełnienie potrzebna jest korekcja wyniku przez dodanie 1 na najmniej znaczącej pozycji dzielnika.

Układ arytmetyczno-logiczny

budowa i działanie układu ALU procesora

Dzielenie liczb binarnych również jest złożoną operacją. W typowym CPU operacja dzielenia zajmuje ponadprzeciętną liczbę cykli.

Dzielenie:
- w kodzie NKB
- w kodzie U2

Wynikiem dzielenia liczb binarnych są iloraz i reszta z dzielenia. Wyznaczenie bitów ilorazu wymaga:
- przesunięcia dzielnika i odjęcie (dodanie dopełnienia) od ilorazu częściowego, jeśli znak wyniku nie ulegnie zmianie – bit ilorazu równy 1,
- wykonanie samego przesunięcia, jeśli odejmowanie powoduje zmianę znaku – bit ilorazu równy 0.

Przykład:

Układ arytmetyczno-logiczny

budowa i działanie układu ALU procesora

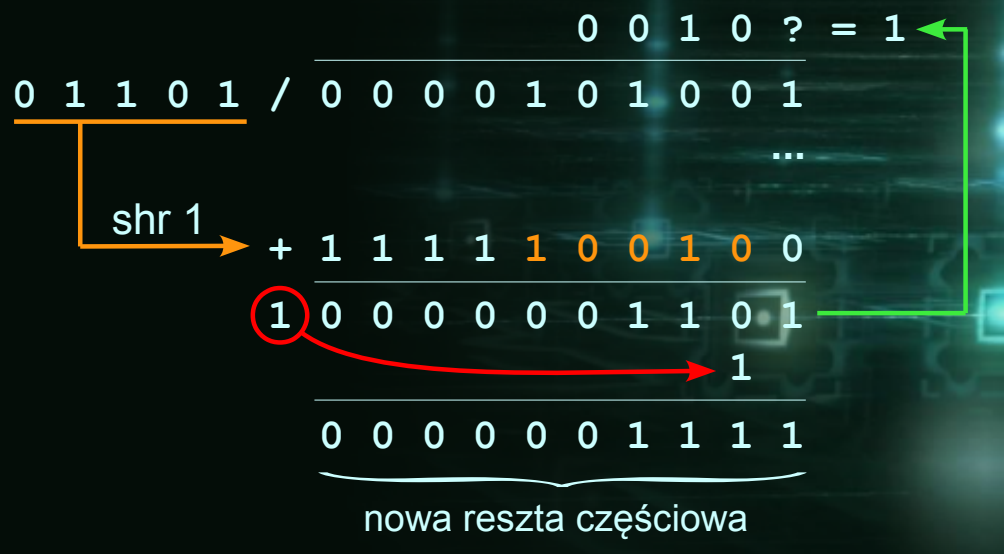
- ALU
- algorytm sumowania
- algorytm mnożenia
- algorytm dzielenia

Dzielenie liczb binarnych również jest złożoną operacją. W typowym CPU operacja dzielenia zajmuje ponadprzeciętną liczbę cykli.

Dzielenie:
- w kodzie NKB
- w kodzie U2

Wynikiem dzielenia liczb binarnych są iloraz i reszta z dzielenia. Wyznaczenie bitów ilorazu wymaga:
- przesunięcia dzielnika i odjęcie (dodanie dopełnienia) od ilorazu częściowego, jeśli znak wyniku nie ulegnie zmianie – bit ilorazu równy 1,
- wykonanie samego przesunięcia, jeśli odejmowanie powoduje zmianę znaku – bit ilorazu równy 0.

Przykład:



Układ arytmetyczno-logiczny

budowa i działanie układu ALU procesora

- ALU
- algorytm sumowania
- algorytm mnożenia
- algorytm dzielenia

Dzielenie liczb binarnych również jest złożoną operacją. W typowym CPU operacja dzielenia zajmuje ponadprzeciętną liczbę cykli.

Dzielenie:
- w kodzie NKB
- w kodzie U2

Wynikiem dzielenia liczb binarnych są iloraz i reszta z dzielenia. Wyznaczenie bitów ilorazu wymaga:
- przesunięcia dzielnika i odjęcie (dodanie dopełnienia) od ilorazu częściowego, jeśli znak wyniku nie ulegnie zmianie – bit ilorazu równy 1,
- wykonanie samego przesunięcia, jeśli odejmowanie powoduje zmianę znaku – bit ilorazu równy 0.

Przykład:

Układ arytmetyczno-logiczny

budowa i działanie układu ALU procesora

- ALU
- algorytm sumowania
- algorytm mnożenia
- algorytm dzielenia

Dzielenie liczb binarnych również jest złożoną operacją. W typowym CPU operacja dzielenia zajmuje ponadprzeciętną liczbę cykli.

Dzielenie:
- w kodzie NKB
- w kodzie U2

W algorytmie dzielenia w U2 należy uwzględnić właściwości kodu. Odpowiednio dla A / B:
- gdy A = -a, B = +b:
  - wpisuje się 1 na pierwszą pozycję ilorazu,
  - dodawana jest liczba ujemna do dodatniej – korekcja wyniku częściowego nie będzie potrzebna.
- gdy A = +a, B = -b:
  - wpisuje się 1 na pierwszą pozycję ilorazu,
  - dzielnik dodaje się do reszty częściowej tylko gdy, w wyniku powstaje przeniesienie, które należy ignorować (wynik nie wymaga korekcji).
- gdy A = -a, B = -b:
  - wpisuje się 0 na pierwszą pozycję ilorazu,
  - dzielnik uzupełniony do 1 oraz wartość 1 (na najmłodszej pozycji dzielnika) dodaje się do reszty częściowej tylko, gdy nie powstaje przeniesienie.

Wynikiem dzielenia liczb binarnych są iloraz i reszta z dzielenia. Wyznaczenie bitów ilorazu wymaga:
- przesunięcia dzielnika i odjęcie (dodanie dopełnienia) od ilorazu częściowego, jeśli znak wyniku nie ulegnie zmianie – bit ilorazu równy 1,
- wykonanie samego przesunięcia, jeśli odejmowanie powoduje zmianę znaku – bit ilorazu równy 0.

Przykład:

Instrukcje zależne od ALU

Instrukcje zależne od ALU

rozkazy CPU w których wykonaniu bierze udział ALU

- operacje logiczne
- przesunięcia bitowe
- instrukcje warunkowe

Jednostka ALU wykonuje różne operacje obliczeniowe. Jednymi z nich są funkcje logiczne, realizowane zgodnie z algebrą Boole'a.

Operacje:
- negacja
- suma logiczna
- iloczyn logiczny
- różnica symetryczna

Instrukcje zależne od ALU

rozkazy CPU w których wykonaniu bierze udział ALU

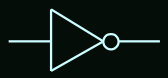
- operacje logiczne
- przesunięcia bitowe
- instrukcje warunkowe

Jednostka ALU wykonuje różne operacje obliczeniowe. Jednymi z nich są funkcje logiczne, realizowane zgodnie z algebrą Boole'a.

Operacje:

- negacja
- suma logiczna
- iloczyn logiczny
- różnica symetryczna

Negacja bitowa polega za zamianie wartości poszczególnych bitów na przeciwną. Tabela funkcji jest następująca:



a	y
0	1
1	0

ab	xy
00	11
01	10
10	01
11	00

Instrukcje zależne od ALU

rozkazy CPU w których wykonaniu bierze udział ALU

- operacje logiczne
- przesunięcia bitowe
- instrukcje warunkowe

Jednostka ALU wykonuje różne operacje obliczeniowe. Jednymi z nich są funkcje logiczne, realizowane zgodnie z algebrą Boole'a.

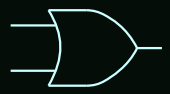
Operacje:
- negacja
- suma logiczna
- iloczyn logiczny
- różnica symetryczna

Negacja bitowa polega za zamianie wartości poszczególnych bitów na przeciwną. Tabela funkcji jest następująca:

a	y
0	1
1	0

ab	xy
00	11
01	10
10	01
11	00

Tabela funkcji jest następująca:



a	b	y
0	0	0
0	1	1
1	0	1
1	1	1

ab	cd	xy
00	00	00
01	10	11
10	10	10
11	00	11

Operandami w ALU są rejestry wielobitowe, w przypadku których operacje logiczne odnoszą się do bitów na tych samy pozycjach. Istnieją również procesory mające możliwość adresowania pamięci bitowo.

Instrukcje zależne od ALU

rozkazy CPU w których wykonaniu bierze udział ALU

- operacje logiczne
- przesunięcia bitowe
- instrukcje warunkowe

Jednostka ALU wykonuje różne operacje obliczeniowe. Jednymi z nich są funkcje logiczne, realizowane zgodnie z algebrą Boole'a.

Operacje:
- negacja
- suma logiczna
- iloczyn logiczny
- różnica symetryczna

Negacja bitowa polega za zamianie wartości poszczególnych bitów na przeciwną. Tabela funkcji jest następująca:

a	y
0	1
1	0

ab	xy
00	11
01	10
10	01
11	00

Tabela funkcji jest następująca:

a	b	y
0	0	0
0	1	1
1	0	1
1	1	1

ab	cd	xy
00	00	00
01	10	11
10	10	10
11	00	11

Tabela funkcji jest następująca:

a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

ab	cd	xy
00	00	00
01	10	00
10	10	10
11	00	00

Niektóre ALU posiadają odpowiednik funkcji AND, której zadaniem jest wyznaczenie wartości znaczników.

Operandami w ALU są rejestry wielobitowe, w przypadku których operacje logiczne odnoszą się do bitów na tych samy pozycjach. Istnieją również procesory mające możliwość adresowania pamięci bitowo.

Instrukcje zależne od ALU

rozkazy CPU w których wykonaniu bierze udział ALU

- operacje logiczne
- przesunięcia bitowe
- instrukcje warunkowe

Jednostka ALU wykonuje różne operacje obliczeniowe. Jednymi z nich są funkcje logiczne, realizowane zgodnie z algebrą Boole'a.

Operacje:
- negacja
- suma logiczna
- iloczyn logiczny
- różnica symetryczna

Tabela funkcji jest następująca:

a	b	y
0	0	1
0	1	0
1	0	0
1	1	1

ab	cd	xy
00	00	11
01	10	00
10	10	11
11	00	00

Tabela funkcji jest następująca:

a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

ab	cd	xy
00	00	00
01	10	00
10	10	10
11	00	00

Niektóre ALU posiadają odpowiednik funkcji AND, której zadaniem jest wyznaczenie wartości znaczników.

Negacja bitowa polega za zamianie wartości poszczególnych bitów na przeciwną. Tabela funkcji jest następująca:

a	y
0	1
1	0

ab	xy
00	11
01	10
10	01
11	00

Tabela funkcji jest następująca:

a	b	y
0	0	0
0	1	1
1	0	1
1	1	1

ab	cd	xy
00	00	00
01	10	11
10	10	10
11	00	11

Operandami w ALU są rejestry wielobitowe, w przypadku których operacje logiczne odnoszą się do bitów na tych samy pozycjach. Istnieją również procesory mające możliwość adresowania pamięci bitiwo.

Instrukcje zależne od ALU

rozkazy CPU w których wykonaniu bierze udział ALU

- operacje logiczne
- przesunięcia bitowe
- instrukcje warunkowe

Przesunięcie bitowe odpowiada operacji mnożenia lub dzielenia przez 2. Pewne CPU potrafią również wykonać przesunięcia cykliczne.

Przesunięcia:
- logiczne
- arytmetyczne
- cykliczne

Instrukcje zależne od ALU

rozkazy CPU w których wykonaniu bierze udział ALU

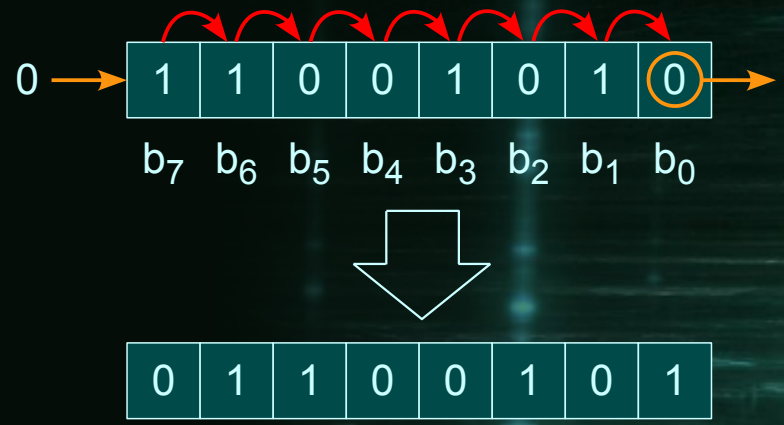
- operacje logiczne
- przesunięcia bitowe
- instrukcje warunkowe

Przesunięcie bitowe odpowiada operacji mnożenia lub dzielenia przez 2. Pewne CPU potrafią również wykonać przesunięcia cykliczne.

Przesunięcia:
- logiczne
- arytmetyczne
- cykliczne

Operacja polega na przemieszczeniu bitów w obrębie rejestru, przy czym znak liczby może nie zostać zachowany.

Przesunięcie w prawo:



Analogicznie wykonywane jest przesunięcie logiczne w lewo, tylko, że bitem wchodzącym jest b_0, a wysuwanym b_7.

Przesunięcie logiczne w lewo odpowiada operacji mnożenia liczby bez znaku przez 2, natomiast przesunięcie w prawo odpowiada operacji dzielenia liczby bez znaku przez 2.

Instrukcje zależne od ALU

rozkazy CPU w których wykonaniu bierze udział ALU

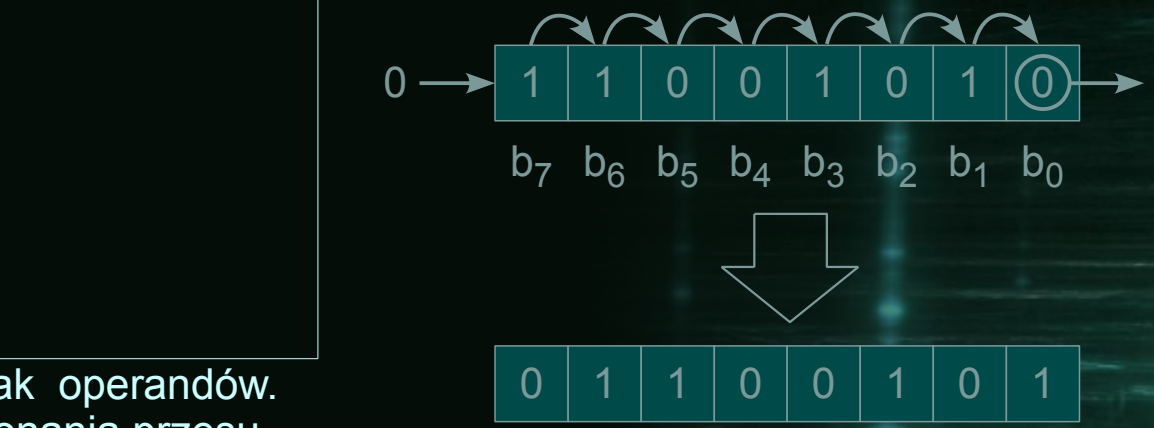
- operacje logiczne
- przesunięcia bitowe
- instrukcje warunkowe

Przesunięcie bitowe odpowiada operacji mnożenia lub dzielenia przez 2. Pewne CPU potrafią również wykonać przesunięcia cykliczne.

Przesunięcia:
- logiczne
- arytmetyczne
- cykliczne

Operacja polega na przemieszczeniu bitów w obrębie rejestru, przy czym znak liczby może nie zostać zachowany.

Przesunięcie w prawo:



Analogicznie wykonywane jest przesunięcie logiczne w lewo, tylko, że bitem wchodzącym jest b_0, a wysuwanym b_7.

Przesunięcie logiczne w lewo odpowiada operacji mnożenia liczby bez znaku przez 2, natomiast przesunięcie w prawo odpowiada operacji dzielenia liczby bez znaku przez 2.

Przesunięcia arytmetyczne zachowują znak operandów. Niektóre CPU nie posiadają możliwości wykonania przesunięć w lewo – zastępowane jest przesunięciem logicznym.

Przykład:

1 0 0 0 1 1 0 0	1 0 0 0 1 1 0 0
1 1 0 0 0 1 1 0	1 0 0 1 1 0 0 0
…	…
1 1 1 1 1 1 1 0	1 1 1 0 0 0 0 0
dzielenie przez 2	mnożenie przez 2

Instrukcje zależne od ALU

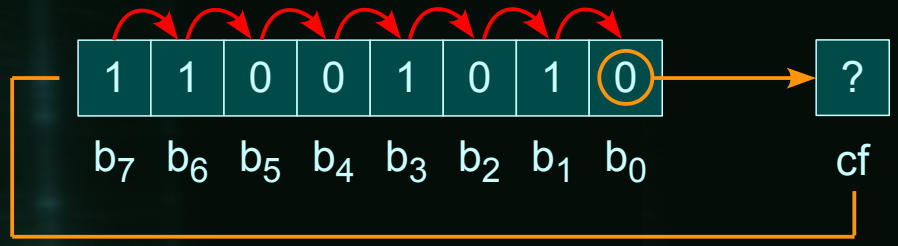
rozkazy CPU w których wykonaniu bierze udział ALU

- operacje logiczne
- przesunięcia bitowe
- instrukcje warunkowe

Przesunięcie bitowe odpowiada operacji mnożenia lub dzielenia przez 2. Pewne CPU potrafią również wykonać przesunięcia cykliczne.

Przesunięcia:
- logiczne
- arytmetyczne
- cykliczne

Przesunięcia cykliczne zachowują wszystkie bity liczby.



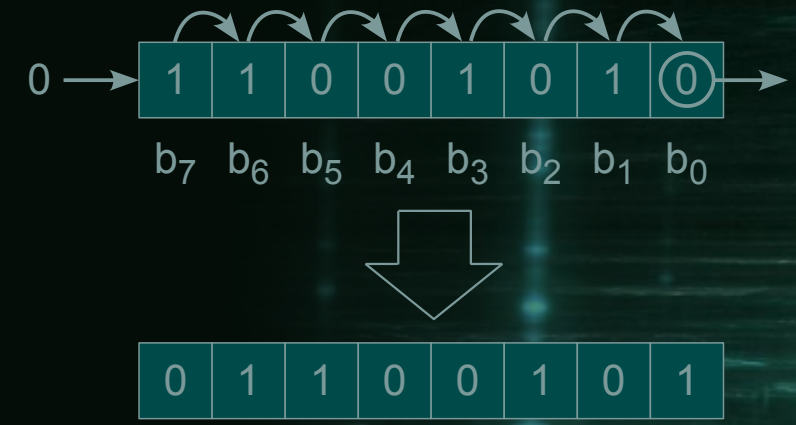
Przesunięcia arytmetyczne zachowują znak operandów. Niektóre CPU nie posiadają możliwości wykonania przesunięć w lewo – zastępowane jest przesunięciem logicznym.

Przykład:

```
1 0 0 0 1 1 0 0    1 0 0 0 1 1 0 0
1 1 0 0 0 1 1 0    1 0 0 1 1 0 0 0
...                ...
1 1 1 1 1 1 1 0    1 1 1 0 0 0 0 0
  dzielenie przez 2    mnożenie przez 2
```

Operacja polega na przemieszczeniu bitów w obrębie rejestru, przy czym znak liczby może nie zostać zachowany.

Przesunięcie w prawo:



Analogicznie wykonywane jest przesunięcie logiczne w lewo, tylko, że bitem wchodzącym jest b_0, a wysuwanym b_7.

Przesunięcie logiczne w lewo odpowiada operacji mnożenia liczby bez znaku przez 2, natomiast przesunięcie w prawo odpowiada operacji dzielenia liczby bez znaku przez 2.

M.Wiśniewski

Instrukcje zależne od ALU

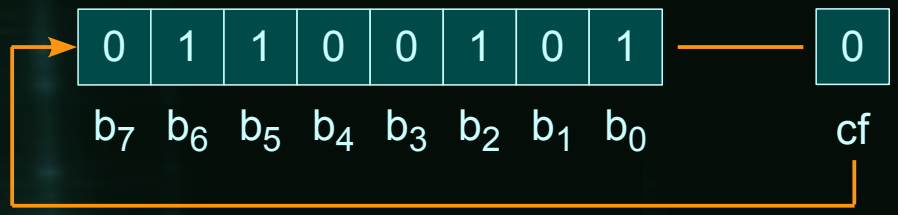
rozkazy CPU w których wykonaniu bierze udział ALU

- operacje logiczne
- przesunięcia bitowe
- instrukcje warunkowe

Przesunięcie bitowe odpowiada operacji mnożenia lub dzielenia przez 2. Pewne CPU potrafią również wykonać przesunięcia cykliczne.

Przesunięcia:
- logiczne
- arytmetyczne
- cykliczne

Przesunięcia cykliczne zachowują wszystkie bity liczby.



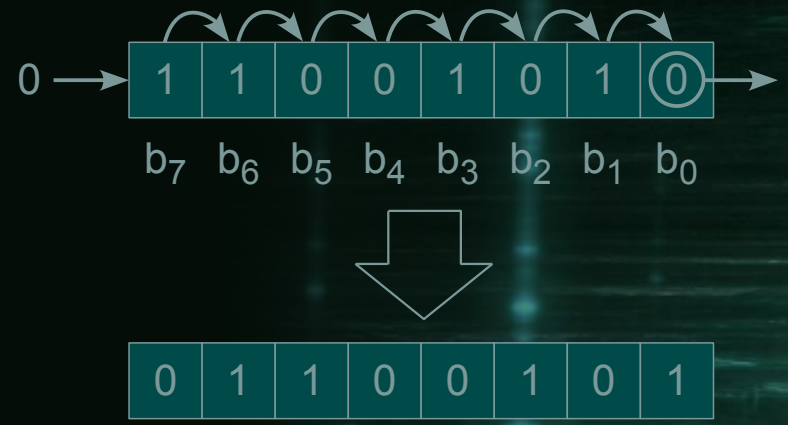
Przesunięcia arytmetyczne zachowują znak operandów. Niektóre CPU nie posiadają możliwości wykonania przesunięć w lewo – zastępowane jest przesunięciem logicznym.

Przykład:

```
1 0 0 0 1 1 0 0    1 0 0 0 1 1 0 0
1 1 0 0 0 1 1 0    1 0 0 1 1 0 0 0
...                ...
1 1 1 1 1 1 1 0    1 1 1 0 0 0 0 0
  dzielenie przez 2    mnożenie przez 2
```

Operacja polega na przemieszczeniu bitów w obrębie rejestru, przy czym znak liczby może nie zostać zachowany.

Przesunięcie w prawo:



Analogicznie wykonywane jest przesunięcie logiczne w lewo, tylko, że bitem wchodzącym jest b_0, a wysuwanym b_7.

Przesunięcie logiczne w lewo odpowiada operacji mnożenia liczby bez znaku przez 2, natomiast przesunięcie w prawo odpowiada operacji dzielenia liczby bez znaku przez 2.

Instrukcje zależne od ALU

rozkazy CPU w których wykonaniu bierze udział ALU

- operacje logiczne
- przesunięcia bitowe
- instrukcje warunkowe

Na podstawie bitów stanu procesor może podejmować decyzje o wykonaniu lub nie wykonaniu pewnych czynności. Takie operacje noszą nazwę warunkowych i mogą obejmować różne grupy rozkazów.

Pojęcia:
- operacje warunkowe
- generowanie warunków

Instrukcje zależne od ALU

rozkazy CPU w których wykonaniu bierze udział ALU

- operacje logiczne
- przesunięcia bitowe
- instrukcje warunkowe

Na podstawie bitów stanu procesor może podejmować decyzje o wykonaniu lub nie wykonaniu pewnych czynności. Takie operacje noszą nazwę warunkowych i mogą obejmować różne grupy rozkazów.

Pojęcia:
- operacje warunkowe
- generowanie warunków

Rozkazy warunkowe mogą obejmować dowolne grupy rozkazów. W niektórych rodzinach procesorów wszystkie rozkazy mogą mieć charakter warunkowy. Taki mechanizm ułatwia implementację konstrukcji programowych typu if-then-else (optymalizacja w aplikacjach mechaniki rozgałęzień w trakcie obliczeń).

Procesory wspierają operacje warunkowe w mniejszym lub większym stopniu:
- architektura x86:
  - CMOVcc – skopiowanie wartości jednego argumentu do drugiego, gdzie cc oznacza rodzaj warunku.
- architektura ARM / procesory VLIW:
  - niemal wszystkie instrukcje.

W procesorach superskalarnych operacje warunkowe zwiększają efektywność spekulatywnego wykonywania rozkazów, jednocześnie wpływają na efektywność przetwarzania potokowego.

Instrukcje zależne od ALU

rozkazy CPU w których wykonaniu bierze udział ALU

- operacje logiczne
- przesunięcia bitowe
- instrukcje warunkowe

Na podstawie bitów stanu procesor może podejmować decyzje o wykonaniu lub nie wykonaniu pewnych czynności. Takie operacje noszą nazwę warunkowych i mogą obejmować różne grupy rozkazów.

Pojęcia:
- operacje warunkowe
- generowanie warunków

Niezależnie od rodzaju operacji, warunki wykonania operacji CPU wyznacza na podstawie bitów stanu, generowanych przez układ ALU. Znaczenie bitów jest następujące:

bit stanu	znak liczby	warunek
o = 1 / o = 0	–	nadmiar / brak nad.
s = 1 / s = 0	–	(-)A / A
z = 1 / z = 0	–	A = B / A ≠ B
c = 1	bez znaku	A < B
c = 0	bez znaku	A ≥ B
c = 1 | z = 1	bez znaku	A ≤ B
c = 0 & z = 0	bez znaku	A > B
s ≠ o	ze znakiem	A < B
s = o	ze znakiem	A ≥ B
z = 1 | s ≠ o	ze znakiem	A ≤ B
z = 0 & s = o	ze znakiem	A > B

Rozkazy warunkowe mogą obejmować dowolne grupy rozkazów. W niektórych rodzinach procesorów wszystkie rozkazy mogą mieć charakter warunkowy. Taki mechanizm ułatwia implementację konstrukcji programowych typu if-then-else (optymalizacja w aplikacjach mechaniki rozgałęzień w trakcie obliczeń).

Procesory wspierają operacje warunkowe w mniejszym lub większym stopniu:
- architektura x86:
  - CMOVcc – skopiowanie wartości jednego argumentu do drugiego, gdzie cc oznacza rodzaj warunku.
- architektura ARM / procesory VLIW:
  - niemal wszystkie instrukcje.

W procesorach superskalarnych operacje warunkowe zwiększają efektywność spekulatywnego wykonywania rozkazów, jednocześnie wpływają na efektywność przetwarzania potokowego.

Superskalarność

Superskalarność

skalarność i superskalarnoćć procesorów

- wstęp
- model procesora

Pierwsze procesory były skalarne, tzn. przetwarzały jeden strumień danych w jednej jednostce czasu.

Pojęcia:
- superskalarność
- ograniczenia
- implementacja

Superskalarność

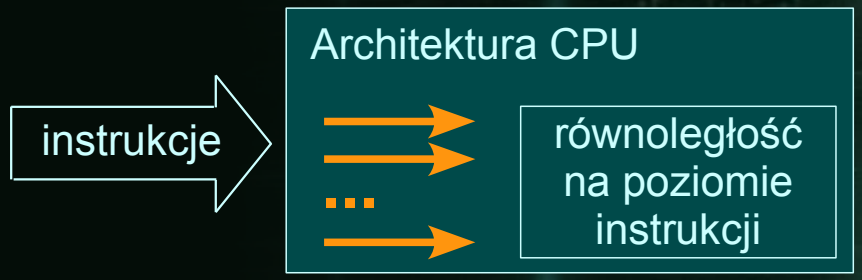
skalarność i superskalarnoćć procesorów

- wstęp
- model procesora

Pierwsze procesory były skalarne, tzn. przetwarzały jeden strumień danych w jednej jednostce czasu.

Pojęcia:
- superskalarność
- ograniczenia
- implementacja

Jest to zdolność procesora do wykonywania więcej niż jednej operacji skalarnej w jednym cyklu zegara.



Superskalarność jest możliwa, gdy:
- procesor posiada odpowiednią architekturę,
- odpowiednie zasoby CPU nie są zajęte,
- instrukcje nie mają wspólnych operandów.

Procesory superskalarne charakteryzuje się:
- wykonywaniem instrukcji poza porządkiem sekwencyjnym programu,
- dynamicznym sprawdzaniem powiązań między danymi rozkazów,
- przetwarzaniem więcej niż jednej instrukcji na cykl.

Superskalarność

skalarność i superskalarnoćć procesorów

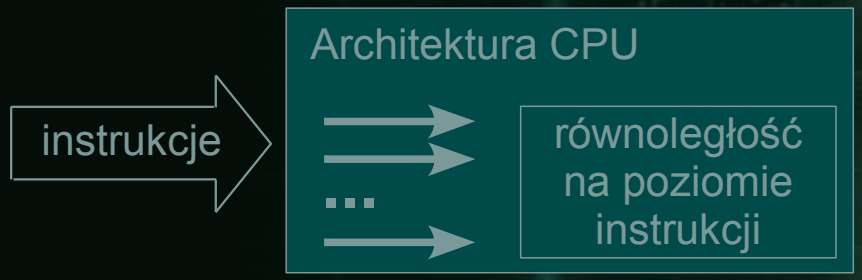
– wstęp

– model procesora

Pierwsze procesory były skalarne, tzn. przetwarzały jeden strumień danych w jednej jednostce czasu.

Pojęcia:
- superskalarność
- ograniczenia
- implementacja

Jest to zdolność procesora do wykonywania więcej niż jednej operacji skalarnej w jednym cyklu zegara.



Superskalarność jest możliwa, gdy:
- procesor posiada odpowiednią architekturę,
- odpowiednie zasoby CPU nie są zajęte,
- instrukcje nie mają wspólnych operandów.

Procesory superskalarne charakteryzuje się:
- wykonywaniem instrukcji poza porządkiem sekwencyjnym programu,
- dynamicznym sprawdzaniem powiązań między danymi rozkazów,
- przetwarzaniem więcej niż jednej instrukcji na cykl.

Zwiększenie wydajności dzięki superskalarności ogranicza:
1. poziom równoległości w strumieniu instrukcji,
2. złożoność i koszt obliczeniowy dispatcher'a oraz logiki analizującej zależności między instrukcjami,
3. mechanika przetwarzania instrukcji rozgałęziających strumień rozkazów.

Superskalarność

skalarność i superskalarnoćć procesorów

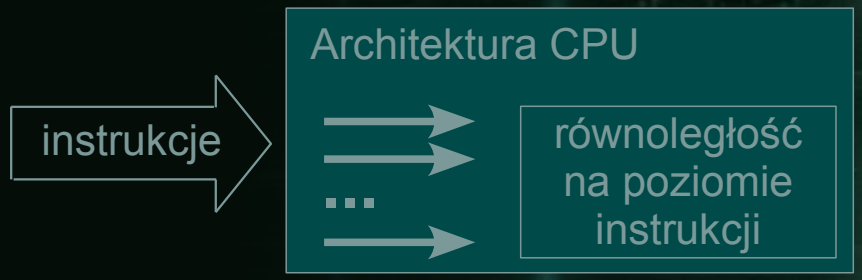
– wstęp

– model procesora

Pierwsze procesory były skalarne, tzn. przetwarzały jeden strumień danych w jednej jednostce czasu.

Pojęcia:
- superskalarność
- ograniczenia
- implementacja

Jest to zdolność procesora do wykonywania więcej niż jednej operacji skalarnej w jednym cyklu zegara.



Implementacja superskalarności wymaga wykorzystania dodatkowych zasobów wykonawczych, takich jak:
- jednostki ALU i/lub inne bloki wykonawcze,
- bufory instrukcji – CPU musi wczytać jednorazowo większą ich liczbę.

Ponadto w procesorach superskalarnych implementuje się:
- mechanizm zmiany kolejności wykonania rozkazów,
- mechanizm spekulatywnego wykonania rozkazów.

Praktycznie wszystkie procesor superskalarne posiadają przetwarzanie potokowe.

Superskalarność jest możliwa, gdy:
- procesor posiada odpowiednią architekturę,
- odpowiednie zasoby CPU nie są zajęte,
- instrukcje nie mają wspólnych operandów.

Procesory superskalarne charakteryzuje się:
- wykonywaniem instrukcji poza porządkiem sekwencyjnym programu,
- dynamicznym sprawdzaniem powiązań między danymi rozkazów,
- przetwarzaniem więcej niż jednej instrukcji na cykl.

Zwiększenie wydajności dzięki superskalarności ogranicza:
1. poziom równoległości w strumieniu instrukcji,
2. złożoność i koszt obliczeniowy dispatcher'a oraz logiki analizującej zależności między instrukcjami,
3. mechanika przetwarzania instrukcji rozgałęziających strumień rozkazów.

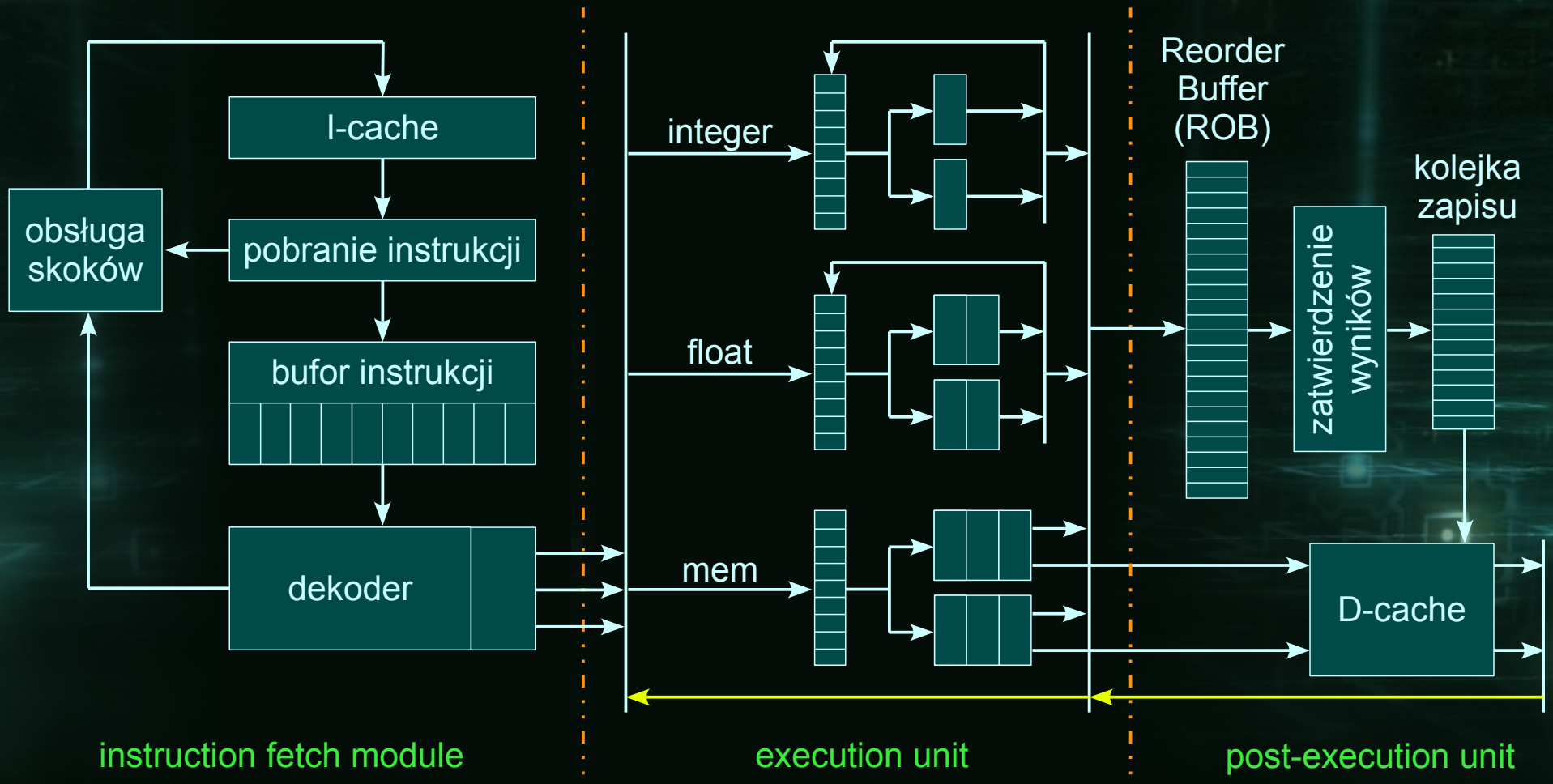
Superskalarność

skalarność i superskalarnoćć procesorów

- wstęp
- model procesora

W architekturze procesorów superskalarnych powinny znajdować się moduły odpowiedzialne za realizację poszczególnych zadań. W szczególności konieczna jest implementacja dispatcher'a oraz pewna reorganizacja jednostek towarzyszących, np. układu pobierania instrukcji.

Model procesora superskalarnego:



Koniec wykładu