1. Liczby ujemne w zapisie binarnym przechowywane są na trzy sposoby:

1. znak – moduł (ZM)
2. zapis uzupełnienie do jednego (ZU1)(U1)
3. zapis uzupełnienie do dwóch (ZU2)(U2)

2. Znak-moduł (ZM)

[https://www.youtube.com/watch?v=ZD0wMdJa-Ns&t=158s](about:blank) (od 2.27 do 9.20)

3. Zapis uzupełnienie do jednego (ZU1)

[https://www.youtube.com/watch?v=ZD0wMdJa-Ns&t=158s](about:blank) (od 9.21 do 13.28)

4. Zapis uzupełnienie do dwóch (ZU2)

[https://www.youtube.com/watch?v=ZD0wMdJa-Ns&t=158s](about:blank) (od 13.30 do 21.20)

5. Przepełnienie (overflow) – czyli liczba nie mieści się w obsługiwanym zakresie.

4 bity: od -8 do 7

8 bitów (1B): od -128 do 127 (char)

8 bitów (1B): od 0 do 255 (unsigned char)

16 bitów (2B): -32678 do 32767 (int)

16 bitów (2B): 0 do 65535 (unsigned int)

6. Podsumowanie:

[https://www.youtube.com/watch?v=ZD0wMdJa-Ns&t=158s](about:blank) (od 21.20)

Zadania 4.docx

Ugruntowanie wiedzy, patrz poniżej.

**Realizacja operacji arytmetycznych w systemach cyfrowych**

W systemach cyfrowych operacje arytmetyczne są wykonywane na liczbach o określonej liczbie bitów. W wyniku realizacji działań arytmetycznych pojawia się wynik, który jest zapamiętywany na określonej liczbie bitów. Na przykład w przypadku dodawania liczb n- bitowych często przyjmuje się, że wynik również zajmuje *n* bitów. Można zauważyć, że wykonanie określonej operacji arytmetycznej na liczbach o określonej liczbie bitów może dać wynik, który wykracza poza zakres wartości liczbowych reprezentowanych w danym kodzie na danej liczbie bitów. Wówczas powstaje nadmiar (ang. *overflow*), który musi być zasygnalizowany użytkownikowi układu cyfrowego. Powstawanie nadmiaru podczas dodawania liczb dodatnich w kodzie NKB ilustruje następujący przykład.

**Przykład. 3.10.** Dodać dwie liczby 4-bitowe w kodzie NKB. Wynik sumowania zapamiętać

na 4 bitach. Liczba *A* = 1100 = 12. Liczba *B* = 1010 = 10.

1100

+1010

------

C = 1 0110 C - przeniesienie

*S* = *A* + *B* = 1100 + 1010 = 10110 = 22. Sprawdzenie 12 + 10 = 22.

W wyniku operacji sumowania pojawia się przeniesienie *c*=1 na pozycję numer cztery. Oznacza to, że zakres wyniku został przekroczony. Rozwiązaniem tego problemu jest wyposażenie układu cyfrowego w sygnalizację przeniesienia lub zwiększenie zakresu wyniku o jeden bit.

W praktyce cyfrowe układy arytmetyczne wykonujące podstawowe działania są wyposażone w sygnał wyjściowy sygnalizujący nadmiar (np. jednostka arytmetyczno-logiczna procesora). Układy te sygnalizują także inne cechy wyniku jak: znak, wynik jest równy zero, wynik zawiera parzystą liczbę jedynek, itp. Bity te są nazywane w układach cyfrowych *znacznikami* (flagami, ang. flags).

* + - 1. **Dodawanie liczb stałopozycyjnych**

**Liczby dodatnie w kodach ZM, U1 i U2 są reprezentowane tak jak w kodzie NKB.** Dodawanie takich liczb wykonuje się w sposób identyczny dla wszystkich kodów. W przypadku liczb A i B dodając bity z pozycji *i*-tej uwzględnia się również przeniesienie z pozycji mniej znaczącej. Dlatego w czasie sumowania na *i*-tej pozycji dodawane są trzy bity: dwa bity *a*i oraz *b*i należące do obu składników oraz przeniesienie *c*i z pozycji mniej znaczącej (*i*-1). Suma *a*i + *b*i + *c*i daje wynik *s*i oraz przeniesienie wychodzące *c*i+1. Zasadę obliczania sumy *s*i oraz przeniesienia *c*i+1 w przypadku sumatora jednobitowego pokazano poniżej.

*a*i *b*i

*c*i+1 Σ *c*i

*s*i

Łącząc szeregowo *n* sumatorów jednobitowych powstanie sumator *n*-bitowy.

*a*i *b*i *c*i *s*i *c*i+1

----------------------------------------

0 0 0 0 0

0 0 1 1 0

0 1 0 1 0

0 1 1 0 1

1 0 0 1 0

1 0 1 0 1

1 1 0 0 1

1 1 1 1 1

Sumowanie dwóch liczb o określonej liczbie bitów może dać wynik, który wykracza poza zakres wartości liczbowych reprezentowanych w danym kodzie na określonej liczbie bitów. Powstaje wówczas nadmiar, który musi być zasygnalizowany przez odpowiedni bit znacznikowy układu cyfrowego.

**Przykład. 3.11.** Dodać dwie, dodatnie liczby 8-bitowe *A*=48 i *B*=58 w kodach NKB, ZM, U1,

U2. Wynik sumowania zapamiętać na 8 bitach.

48 00110000 NKB, ZM, U1, U2

+58 + 0111010

----- ---------------

106 01101010

Jest to poprawny wynik dodawania liczb *A*=48 i *B*=58.

**Przykład. 3.12.** Dodać dwie, dodatnie liczby 8-bitowe *A*=78 i *B*=57 w kodach NKB, ZM, U1, U2. Wynik sumowania zapamiętać na 8 bitach.

1111

78 01001110 NKB,ZM,U1,U2

+57 +00111001

----- --------------

135 10000111

W kodzie NKB otrzymana suma jest równa 135, a więc wynik jest poprawny. Natomiast w kodach ZM, U1 i U2, ze względu na ustawiony bit znaku na najstarszej pozycji, otrzymany wynik jest ujemny. Rzeczywiście liczby 135 nie da się przedstawić na 7 bitach. Prawidłowe dodawanie takich liczb można wykonać jedynie na większej liczbie bitów. W rozpatrywanym przypadku wystarczy wykonać dodawanie na słowach 9-bitowych.

1111

78 001001110 NKB,ZM,U1,U2

+57 +000111001

----- ---------------

135 010000111 = 135

* + - 1. **Odejmowanie liczb stałopozycyjnych**

Odejmowanie liczb w kodach uzupełnieniowych U1, U2 można sprowadzić do dodawania dwóch argumentów o przeciwnych znakach.

W kodzie ZM odejmowanie dwóch liczb o przeciwnych znakach sprowadza się do porównania modułów liczb w celu ustalenia znaku wyniku jako znaku liczby o większym module. Następnie wyznacza się różnicę modułów przez odjęcie mniejszego modułu od większego modułu w sposób identyczny jak dla liczb w kodzie NKB.

W przypadku odejmowania od *a*i = 0 wartości *b*i = 1 należy pobrać pożyczkę *p*i+1 = 1 wychodzącą z pozycji (*i*+1). Wartość tej pożyczki na pozycji (*i*) wynosi 2. Podczas odejmowania bitów na *i*-tej pozycji należy odjąć również pożyczkę *p*i wchodzącą z pozycji mniej znaczącej (*i*-1). Wzór na odejmowanie bitów z pozycji *i*-tej ma następującą postać:

*d*i = *a*i - *b*i - *p*i + 2∗*p*i+1 .

Np. Odejmowanie *A*=2 i *B*=1. Różnica *D*=*A*-*B*=1. W tym przypadku *p*0=0, *p*1=1 i *p*2=0.

W wyniku odejmowania *d*0 = *a*0 - *b*0 - *p*0 + 2∗*p*1 = 0 – 1 – 0 + 2∗1 = 1 oraz *d*1 = *a*1 – *b*1 – *p*1 +

2∗*p*2 = 1 – 0 – 1 + 2∗0 = 0.

2

10 (2)

- 01 (1)

-------

01 (1)

Poniżej przedstawiono sposób obliczania różnicy dwóch liczb *d*i = *a*i - *b*i - *p*i + 2∗*p*i+1 z uwzględnieniem pożyczki.

*a*i *b*i *p*i *d*i *p*i+1

----------------------------------------

0 0 0 0 0

0 0 1 1 1

0 1 0 1 1

0 1 1 0 1

1 0 0 1 0

1 0 1 0 0

1 1 0 0 0

1 1 1 1 1

**Przykład. 3.13.** Odjąć w kodzie ZM dwie liczby 8-bitowe *A*=-28 i *B*=10.

W kodzie ZM odejmowanie realizowane jest na modułach liczb a następnie uzupełniany jest znak wyniku. Porównanie modułów liczb prowadzi do przyjęcia ujemnego znaku wyniku.

Odejmując od 28 liczbę 10 otrzymuje się:

00011100 = 28

- 00001010 = 10

------------------------

00010010 = 18

Uwzględniając znak wyniku otrzymana różnica w kodzie ZM ma wartość –18 = 10010010.

Odejmowanie w kodzie ZM wymaga ustalenia znaku wyniku na podstawie porównania modułów argumentów. Wynika stąd, że w kodzie ZM odejmowanie realizuje się inaczej niż dodawanie w tym kodzie, które nie wymaga takiego porównywania. Jest to wada kodu ZM, która powoduje, że jest on rzadko stosowany.

W przypadku kodów uzupełnieniowych U1 i U2 przedstawiona wada kodu ZM nie występuje. Odejmowanie wykonuje się w tych kodach w taki sam sposób jak dodawanie z uwzględnieniem zakresu liczb dodatnich i ujemnych w obu typach kodów.

***Odejmowanie w kodzie U1***

W kodzie U1 wartość dziesiętną liczby całkowitej R = an-1an-2...a1a0 można obliczyć ze wzoru:

Dla liczb dodatnich (an-1=0). Dla liczb ujemnych (an-1=1), natomiast na pozostałych (*n*-1) bitach znajduje się liczba dodatnia *X* będąca dopełnieniem modułu liczby *R* do 2(*n*-1) –1 (suma |*R*| + *X* wynosi 2(*n*-1) -1). Dla *R* ujemnego wartość (n-1)-bitowej liczby *X* można obliczyć ze

wzoru:

.

Sposób odejmowania zależy od wartości liczby X oraz znaków odejmowanych liczb.

**Przykład. 3.14.** Przedstawić w 8-bitowym kodzie U1 liczbę *R*=-20. Podać wartość liczby *X*

zdefiniowanej na pierwszych siedmiu bitach słowa kodowego. Obliczyć wartość *W* = |*R*| + *X*.

-*R* = 20 = 0001 0100 *R* = -20 = 1 1101011

*X* = 1101011 = 127 + (– 20) = 107 *W* = |*R*| + *X* = 127.

**Odejmowanie liczb o różnych znakach**

Dodając do ujemnej liczby *A* (*X* = 2n-1-1 + *A*) dodatnią liczbę *B* (*Y* = *B*) obliczana jest suma bitów obu liczb z pozycji od 0 do *n*-2 (*X* + *Y*), która wynosi

(2*n*−1 −1) + *A* + *B* .

Jeśli moduł liczby ujemnej jest większy (równy) niż moduł liczby dodatniej, tj. |*A*| ≥ *B*, to suma *A*+*B* jest ujemna (równa zero) i wynik jest mniejszy (równy) od 2(*n*-1)-1. W tym przypadku nie następuje przeniesienie na pozycję (*n*-1) o wadze 2(*n*-1). Na najstarszej pozycji pozostaje bit znaku (an-1=1), a otrzymana liczba jest ujemna. Przedstawiony przypadek ilustruje następujący przykład.

**Przykład. 3.15.** Dodać w 8-bitowym kodzie U1 liczby *A*=-20 i *B*=7.

1111

11101011 (-20)

+00000111 (+7)

-------------

11110010 (-13)

00001101 (+13)

W tym przypadku |*A*| > *B*. Nie powstało przeniesienie na pozycję najbardziej znaczącą (*n*-1)

i wynik jest poprawny.

W przypadku dodawania w kodzie U1 liczb *n*-bitowych *A* i *B* o przeciwnych znakach nieprawidłowy wynik pojawi się, gdy moduł liczby dodatniej jest większy od modułu liczby ujemnej, tj. gdy *B* > *|A*|. W tym przypadku na (*n*-1) bitach liczby ujemnej *A*, zapisanej w kodzie U1, zakodowana jest liczba dodatnia *X* będąca dopełnieniem modułu liczby do 2(*n*-1) -1, natomiast na odpowiednich (*n*-1) pozycjach liczby dodatniej *B* jest zakodowany jej moduł. Jeśli wartość modułu liczby *B* jest większa niż modułu *A*, to po dodaniu pierwszych (*n*-1) bitów pojawi się liczba większa od 2(*n*-1)–1. Spowoduje to pojawienie się przeniesienia na pozycję bitu znaku co prowadzi do zmiany bitu wyniku na 0 (wynik ma być dodatni). Wartość liczbowa wyniku jest równa sumie bitów obu liczb z pozycji od 0 do *n*-2, pomniejszonej o wartość przeniesienia na pozycję (*n*-1). Stąd, suma liczb ma wartość   
(2*n*−1 −1) + *A* + *B* − 2*n*−1 = *A* + *B* −1. Jest to wartość nieprawidłowa. W takim przypadku należy wykonać krok korekcyjny i dodać do wyniku 1. Przedstawione rozważania ilustruje następujący przykład.

**Przykład. 3.16.** Dodać w 8-bitowym kodzie U1 liczby *A*=-20 i *B*=45.

11 1111

11101011 (-20)

+00101101 (+45)

-------------

C=1 00011000 (+24)

W tym przypadku *B* >*|A*|. Powstało przeniesienie na pozycję bitu znaku. W wyniku dodawania otrzymano liczbę +24 oraz przeniesienie. Krok korekcyjny polega na dodaniu jedynki do wyniku:

00011000 (+24)

+ 00000001 (+1) korekcja

--------------

00011001 (+25)

Konieczność dokonywania korekcji wyniku w przypadkach powstawania przeniesienia na najbardziej znaczącą pozycję jest wadą kodu U1, gdyż wydłuża czas dodawania. W skrajnym przypadku dodanie jedynki w ramach korekcji wyniku może spowodować, że przeniesienie będzie propagować przez wszystkie pozycje wyniku. Może to prowadzić do dwukrotnego zwiększenia czasu dodawania. Wady tej nie ma kod U2, w którym nie jest wymagana korekcja podczas dodawania liczb z przeciwnym znakiem.

**Odejmowanie liczb ujemnych**

Dodawanie liczb ujemnych *A* i *B* w kodzie U1 zawsze da przeniesienie wychodzące z pozycji (*n*-1), gdyż oba bity znaku są jedynkami. Jeśli suma bitów pochodzących z pozycji od 0 do (*n*-2) (*X* + *Y*)obu liczb jest większa lub równa 2(*n*-1)-1, to aby otrzymać prawidłowy wynik należy wykonać krok korekcyjny dodając do wyniku 1.

**Przykład. 3.17.** Dodać w 8-bitowym kodzie U1 liczby *A*=-20 i *B*=-30.

1 1 11

1 1101011 (-20) (107)

+1 1100001 (-30) (97)

--------------

C=1 1 1001100 (-51)

0 0000001 (+1) korekcja

-------------

1 1001101 (-50)

0 0110010 (50)

Suma liczb (*n*-1)-bitowych jest większa niż 127 (107 + 97 = 204). W celu otrzymania prawidłowej wartości do wyniku dodano jeden.

**Przykład. 3.18.** Dodać w 8-bitowym kodzie U1 liczby *A*=-85 i *B*=-42.

1 0101010 (-85) (42)

+1 1010101 (-42) (85)

---------------

C=1 0 1111111 (+127)

0 0000001 (+1) korekcja

---------------

1 0000000 (-127)

Suma liczb (*n*-1)-bitowych jest równa 127 (42 + 85). W celu otrzymania prawidłowej wartości do wyniku dodano jeden.

Jeśli suma bitów pochodzących z pozycji od 0 do (*n*-2) (*X*+*Y*) obu liczb jest mniejsza niż 2(*n*-1)-1, to dodawanie liczb, nawet po uwzględnieniu korekcji, prowadzi do nieprawidłowegowyniku.

**Przykład. 3.19.** Dodać w 8-bitowym kodzie U1 liczby *A*=-30 i *B*=-98.

1

1 1100001 (-30) (97)

+1 0011101 (-98) (29)

---------------

C=1 0 1111110 (+126)

0 0000001 (+1) korekcja

--------------

0 1111111 (+127)

Suma liczb (*n*-1)-bitowych jest mniejsza niż 127 (97 + 29 = 126). Nawet po dodaniu jedynki nie otrzymano prawidłowego wyniku (*A*+*B* =–128). Jeśli jednak wykona się to samo dodawaniedla liczb 9-bitowych (*n*=9), to otrzyma się przypadek, w którym suma liczba (*n*-1) bitowych przekroczy 2(*n*-1)-1 = 255. Wówczas wynik będzie prawidłowy po dodaniu jedynki.

1 11100001 (-30) (225)

+1 10011101 (-98) (157)

-----------

C=1 1 01111110 (-129)

0 00000001 (+1) korekcja

-----------

1 01111111 (-128)

Konieczność wykonywania kroku korekcyjnego w przypadku, gdy na pozycji najbardziej znaczącej (*n*-1) pojawia się przeniesienie, jest wadą kodu U1. Realizacja operacji odejmowania w kodzie U2 nie ma tej wady.

***Odejmowanie w kodzie U2***

W kodzie U2 wartość dziesiętną liczby całkowitej R = an-1an-2...a1a0 można obliczyć ze wzoru:

Dodając do ujemnej liczby *A* dodatnią liczbę *B* obliczana jest suma bitów obu liczb z pozycji od 0 do *n*-2, która wynosi

(2*n*−1 ) + *A* + *B* .

Jeśli suma ta przekroczy wartość 2(*n*-1) –1, to pojawi się przeniesienie na pozycję bitu znaku

(pozycja o numerze *n*-1). Znak wyniku będzie wówczas równy 0 a wartość sumy wyniesie

(2*n*−1 ) + *A* + *B* −2*n*−1= *A* + *B* .

Wynika stąd, że dodawanie liczb z przeciwnym znakiem w kodzie U2 daje poprawny wynik

bez kroku korekcyjnego.

**Przykład. 3.20.** Dodać w 8-bitowym kodzie U2 liczby *A*=-20 i *B*=45.

11 11

11101100 (-20)

+00101101 (+45)

--------------

C=1 00011001 (+25)

Wynik dodawania jest poprawny i wynosi -20 + 45 = 25.

**Przykład. 3.21.** Dodać w 8-bitowym kodzie U2 liczby *A*=20 i *B*=-45.

1

00010100 (+20)

+11010011 (-45)

--------------

11100111 (-25)

00011001 (+25)

Wynik dodawania jest poprawny i wynosi 20 - 45 = -25.

Dodając dwie liczby ujemne można przekroczyć dopuszczalny zakres dla kodu U2 i wówczas na najbardziej znaczącej pozycji powstanie zero, co oznacza, że wynik jest dodatni. Nie jest to zgodne z prawdą, gdyż obie liczby są ujemne. W takim przypadku należy wykonać operacje na liczbach o większej liczbie bitów.

**Przykład. 3.22.** Dodać w 8-bitowym kodzie U2 liczby *A*=-20 i *B*=-45.

1

11101100 (-20)

+11010011 (-45)

--------------

C=1 10111111 (-65)

Wynik dodawania jest poprawny i wynosi -20 -45 = -65.

**Przykład. 3.23.** Dodać w 8-bitowym kodzie U2 liczby *A*=-40 i *B*=-120.

11

11011000 (-40)

+10001000 (-120)

------------

C=1 01100000 (+96)

W przypadku uwzględniania 8 bitów wynik nie jest prawidłowy. Jeśli uwzględni się liczby 9-bitowe, to otrzymany wynik będzie poprawny.

1 11

111011000 (-40)

+110001000 (-120)

-------------

C=1 101100000 (-160)

010100000 (+160)

* + - 1. **Mnożenie liczb stałopozycyjnych**

W systemie dwójkowym liczby *A* i *B* mnoży się w tylu krokach z ilu bitów składają się czynniki. Jeśli mnoży się *pisemnie* dwie liczby *n*-bitowe w kodzie NKB, to wykonuje się to w *n* krokach. W każdym kroku mnoży się jedną cyfrę mnożnika *B* przez mnożną *A*. W wyniku otrzymuje się iloczyn cząstkowy, który jest wpisywany z odpowiednim przesunięciem. Na końcu algorytmu sumowanych jest *n* iloczynów cząstkowych. Wynik mnożenia może byćco najwyżej (*n*+*n*) – bitowy.

**Przykład. 3.24.** Pomnożyć pisemnie w kodzie NKB 4-bitowe liczby binarne *A* = 0101 = 5 i *B* = 0111 = 7. Iloczyn *A* \* *B* = 35.

0101 mnożna (+5)

\*0111 mnożnik (+7)

---------

111 przeniesienie

--------------

0101

0101

0101

+0000

----------

0100011 iloczyn (+35)

* + - 1. **Dzielenie liczb stałopozycyjnych**

W systemie dwójkowym dzielenie liczby *A* przez *B* można sprowadzić do odejmowania dzielnika *B* od dzielnej *A*. Operacja dzielenia pisemnego składa się z kilku kroków odejmowania.W wyniku dzielenia liczb dwójkowych w kodzie NKB otrzymuje się część całkowitą i resztę, którą można rozwinąć w ułamek z określoną dokładnością.

**Przykład. 3.27.** Podzielić pisemnie w kodzie NKB liczbę binarną *A* = 01000010 = 66 i *B* = 1100 = 12.

101 (+5)

---------

01000010 : 1100

- 1100

-------------

010010

- 1100

-------------

00110 (+6) reszta

Wynik działania *A* : *B* = 66 : 12 = 5 reszta 6, gdyż 66 = 5∗12 + 6.

1. **Zadania do wykonania**
2. Podstawowe działania w systemie dwójkowym.
3. Odtwórz mnożenie, w którym pewne cyfry dwójkowe zastąpiono krzyżykami.

x x x 1 1 x

x x x x

x x 0 x x x

x 1 x x x x

x x x 1 x x x x 1

1. Odtwórz dzielenie, w którym pewne cyfry dwójkowe zastąpiono krzyżykami.

x x 1 x x x x x : x x x = x x x x x x

x x x

x 1 x

x x x

x x x

x x x

= = =

W poniższym dzieleniu w systemie dwójkowym odtwórz położenie cyfr 0 i 1.

1xx1x1: 1x = 1xx1

x1

x1

xx

==x1

x1

==

1. Odjąć pisemnie liczby binarne w kodzie NKB o wartościach *A* = 14.5 i *B* = 7.625.
2. Pomnożyć pisemnie liczby binarne w kodzie NKB o wartościach *A* = 13.25 i *B* = 5.
3. Podzielić pisemnie liczby binarne w kodzie NKB o wartościach *A* = 19.75 i *B* = 4.125. Wyznaczyć iloraz i resztę.
4. Dodać w kodzie NKB następujące liczby binarne 8-bitowe o wartościach *A* = 94 i *B* = 28.
5. Odjąć w kodzie ZM liczby binarne 8-bitowe o wartościach *A* = +12 i *B* = -25.
6. Dodać w kodzie U1 liczby binarne 8-bitowe o wartościach *A* = -13 i *B* = 46.
7. Dodać w kodzie U1 liczby binarne 8-bitowe o wartościach *A* = -25 i *B* = -99.
8. Dodać w kodzie U2 liczby binarne 9-bitowe o wartościach *A* = -35 i *B* = -135.
9. Wykonaj te dodawania bezpośrednio w kodzie U2 oraz podaj w systemie dziesiętnym wartości dodawanych liczb i także wartość wyniku w systemie dziesiętnym. Zinterpretuj wynik pod kątem ew. nadmiaru
10. 10101110 + 01011101
11. 01011010 + 01001111
12. 10011010+10001111
13. 01111101+10111100