ASEMBLER 6502
(errata)

Opracowanie:
Mariusz "Mario" Bielak

Przyjęta forma dokumentu:

Pole **CZARNE**	**– Informacja o lokalizacji błędu**
Pole **ZIELONE**	– Opis błędu
Pole **SZARE**	– Wyciąg z oryginału, błędy zaznaczane są na **czerwono na żółtym tle**
Pole **BIAŁE**	– Tekst poprawiony, poprawki zaznaczane są na **zielono na żółtym tle**

Wykaz błędów (poprawionych):

Strona 4, linia 5

Brak znaku przeniesienia wyrazu do nowej linii

```
nia w asemblerze oraz jego specyficznymi właściwościami. Zna
```

```
nia w asemblerze oraz jego specyficznymi właściwościami. Zna-
```

Strona 8, linia 19

Błędne odwołanie do nieistniejącego podpunktu

```
samym programie. O stosowaniu etykiet powiemy szerzej w punk-
cie 1.6.2.
```

```
samym programie. O stosowaniu etykiet powiemy szerzej w punk-
cie 1.7.3.
```

Strona 9, linia 18

Niepotrzebny zaimek w zdaniu

```
     Ważnym udogodnieniem stało się to, że gdy pojedyńcze roz-
kazy JM realizują się w większości wąskie, cząstkowe zadania,
```

```
     Ważnym udogodnieniem stało się to, że gdy pojedyńcze roz-
kazy JM realizują w większości wąskie, cząstkowe zadania,
```

Strona 11, linia 28

Literówka

```
także wykrywanie i usuwanie błędów. Pamiętajmy zarazem: nie
ma języków lepszych i gorszych i nie ma takiego, którym miał-
by same tylko zalety. Wybór języka zależy przede wszystkim
```

```
także wykrywanie i usuwanie błędów. Pamiętajmy zarazem: nie
ma języków lepszych i gorszych i nie ma takiego, który miał-
by same tylko zalety. Wybór języka zależy przede wszystkim
```

Strona 14, linia 6

Literówka

```
     Dlaczego wybrano taką właśnie formę reprezentacji danych,
z pozoru niewygodną ze względu na nasze przezwyczajenia do
posługiwania się układem dziesiętnym (dec.)? Kod binarny zna-
```

```
     Dlaczego wybrano taką właśnie formę reprezentacji danych,
z pozoru niewygodną ze względu na nasze przyzwyczajenia do
posługiwania się układem dziesiętnym (dec.)? Kod binarny zna-
```

Strona 22, linia 6

Literówka

```
     Etykiety stanowią bardzo znaczne udogodnienie. Wykorzys-
tują się je na dwa sposoby. Pierwszy polega na tym, że na po-
```

```
     Etykiety stanowią bardzo znaczne udogodnienie. Wykorzys-
tuje się je na dwa sposoby. Pierwszy polega na tym, że na po-
```

Strona 22, linie 20-22

Błędnie wyliczone adresy. W linii 20 rozkaz "JSR SKOK" zajmuje 3 bajty (pod adresami $600, $601 i $602), zatem kolejny rozkaz BRK powinien znajdować się pod adresem $603 (a nie $602), a oznaczony etykietą SKOK rozkaz RTS pod adresem $604. Taki też adres (z odwróconą kolejnością bajtów) powinien znajdować się jako operand dla JSR.

```
                          Adres       Kod
     JSR SKOK             0600        20 03 06
     BRK                  0602        00
SKOK RTS                  0603        60
```

```
                          Adres       Kod
     JSR SKOK             0600        20 04 06
     BRK                  0603        00
SKOK RTS                  0604        60
```

Strona 25, linia 6

Urwany wyraz przy przenoszeniu do nowej linii

```
w którym jest teraz 37. Kolejny rozkaz, STA M, sprawia, że rów-
komórce o adresie M zapisane zostaje 37. Dwa ostatnie rozkazy
zwiększają tę wartość o 2.
```

```
w którym jest teraz 37. Kolejny rozkaz, STA M, sprawia, że rów-
nież w komórce o adresie M zapisane zostaje 37. Dwa ostatnie
rozkazy zwiększają tę wartość o 2.
```

Strona 26, linia 7

Literówka

```
     Jest jednak naturalne, że chcielibyćmy zobaczyć, czy
```

```
     Jest jednak naturalne, że chcielibyśmy zobaczyć, czy
```

Strona 27, linia 17

Błędny kod hex rozkazu RTS

```
060B      RTS             06
```

```
060B      RTS             60
```

Strona 29, linia 16

Niepotrzebny przyimek (lub próba przenoszenia wyrazu z następnej linii)

```
     x 2. Napiszmy w asemblerze w hex program odpowiadający na
następującemu ciągowi instrukcji w Basicu (pamiętajmy o kon-
```

```
     x 2. Napiszmy w asemblerze w hex program odpowiadający
następującemu ciągowi instrukcji w Basicu (pamiętajmy o kon-
```

Strona 30, linia 2 (od spodu)

Dwa różne systemy objaśnień na rysunku (cyfry) i w podpisie pod rysunkiem (litery)

```
     Rys. 2.1  Architektura typowego mikrokomputera. Objaśnie-
               nia: A - szyna danych, B - szyna adresów, C -
               linia sterowania.
```

```
     Rys. 2.1  Architektura typowego mikrokomputera. Objaśnie-
               nia: 1 - szyna danych, 2 - szyna adresów, 3 -
               linia sterowania.
```

Strona 33, linia 13

Niepotrzebny przyimek (lub próba przenoszenia wyrazu z następnej linii)

```
zeruje RAM, wprowadza do wielu komórek sterujących wartości po
początkowe, sprawdza, czy są przyłączone urządzenia zewnętrz-
```

```
zeruje RAM, wprowadza do wielu komórek sterujących wartości
początkowe, sprawdza, czy są przyłączone urządzenia zewnętrz
```

Strona 37, linia 14
Literówka

```
300 nanosekund czyli 300 milardowych części sekundy. Do swo-
```

```
300 nanosekund czyli 300 miliardowych części sekundy. Do swo-
```

Strona 39, linia 14
Odwołanie do niewłaściwego numeru rysunku

```
     X i Y - dwa ostatnie rejestry przedstawione na rys. 2.1
```

```
     X i Y - dwa ostatnie rejestry przedstawione na rys. 2.2
```

Strona 39, linia 27
Brak znaku przeniesienia wyrazu do nowej linii

```
     Rola rejestrów indeksowych będzie omówiona przy odpowie
dnich trybach adresowania. Pamiętać trzeba, że ich funkcje
```

```
     Rola rejestrów indeksowych będzie omówiona przy odpowie-
dnich trybach adresowania. Pamiętać trzeba, że ich funkcje
```

Strona 41, linia 6
Literówka

```
poprzdniego, lecz już w czasie jego wykonywania. Jest to rów-
```

```
poprzednio, lecz już w czasie jego wykonywania. Jest to rów-
```

Strona 42, linia 11
Literówka

```
go opisu. A trwało to tym razem dwie milionowe częci sekundy.
```

```
go opisu. A trwało to tym razem dwie milionowe części sekundy.
```

Strona 43, linia 7
Literówka

```
stanie jest absolutnie koniecznie w pięciu ważnych trybach ad-
```

```
stanie jest absolutnie konieczne w pięciu ważnych trybach ad-
```

Strona 43, linia 13
Literówka

```
najczęciej wykorzystywane dane, w tym adresy.
```

```
najczęściej wykorzystywane dane, w tym adresy.
```

Strona 47, linia 3 (od spodu)
Niepotrzebny zaimek w zdaniu

```
     Tę ważną właściwość liczb binarnych wykorzystują się
wszystkie mikroprocesory, a wśród nich 6502. Ma on specjalne
```

```
     Tę ważną właściwość liczb binarnych wykorzystują
wszystkie mikroprocesory, a wśród nich 6502. Ma on specjalne
```

Strona 48, linie 27
Literówka

```
albo za nas komputer musimy poznać skuteczne metody konwersji
```

```
albo za nas komputer musimy poznać skuteczne metody konwersji
```

Strona 49, linia 5
Literówki

```
     Najbliższe tej różnicy jest liczby 2048 czyli 2^11. Zapi-
```

```
     Najbliższa tej różnicy jest liczba 2048 czyli 2^11. Zapi-
```

Strona 50, linia 16

Błędny operator

```
c/ 4+C=1  z przeniesienia. C hex = 12 dec. Razem 17 dec czyli
          11 hex. Powstało przeniesienie. Wpisujemy 1.
```

```
c/ 4+C+1  z przeniesienia. C hex = 12 dec. Razem 17 dec czyli
          11 hex. Powstało przeniesienie. Wpisujemy 1.
```

Strona 54, linia 13

Literówka

```
stawionymi bajtami znajduje się pod adresami E i E+1, a druga
- pod adresam F i F+1. Zastosujemy wówczas tryb adresowania
```

```
stawionymi bajtami znajduje się pod adresami E i E+1, a druga
- pod adresami F i F+1. Zastosujemy wówczas tryb adresowania
```

Strona 55, linia 19

Zdublowany wyraz

```
nia do dwóch. Powstały zatem dwa kody dla przedstawienia liczb
liczb: jeden dla liczb bez znaku, a drugi - ze znakiem. Okaza-
```

```
nia do dwóch. Powstały zatem dwa kody dla przedstawienia
liczb: jeden dla liczb bez znaku, a drugi - ze znakiem. Okaza-
```

Strona 55, linia 23

Urwany wyraz przy przenoszeniu do nowej linii

```
     Czym jest ów kod uzupełnienia do dwóch zwany również ko-
krócej kodem uzupełnieniowym? Jego zasadę można poglądowo wy-
```

```
     Czym jest ów kod uzupełnienia do dwóch zwany również
krócej kodem uzupełnieniowym? Jego zasadę można poglądowo wy-
```

Strona 57, linie 28 i 29

Niepotrzebna cyfra (linia 28) oraz znak dwukropka (linia 29)

```
   00000111    5+7
+  00001100    +12.
   00010011    +19          C=0    V=0
```

```
   00000111     +7
+  00001100    +12
   00010011    +19          C=0    V=0
```

Strona 58, linia 6

Błędna reprezentacja wartości -4 w kodzie U2

```
   11111110    -2
+  11111010    -4
1  11111010    -6         C=1     V=0
```

```
   11111110    -2
+  11111100    -4
1  11111010    -6         C=1     V=0
```

Strona 59, linie 18 i 19

Literówka (w linii 18) i błędny numer podrozdziału (w linii 19)

```
przeniesienia . Pozy tym programy są analogiczne jak przy do-
dawaniu, przedstawione w punkcie 5.3. Oto ich wersje:
```

```
przeniesienia . Poza tym programy są analogiczne jak przy do-
dawaniu, przedstawione w punkcie 3.5. Oto ich wersje:
```

Strona 61, linia 12

Niepotrzebny przyimek (lub próba przenoszenia wyrazu z następnej linii)

```
     W celu odwzorowania pętli i tablic w asemblerze trzeba za
zastosować rozkazy, których jeszcze nie poznaliśmy, toteż od-
```

```
     W celu odwzorowania pętli i tablic w asemblerze trzeba
zastosować rozkazy, których jeszcze nie poznaliśmy, toteż od-
```

Strona 61, linia 19
Błędna wartość argumentu rozkazu LDX (Uzasadnienie: patrz następne 5 linii)

```
     LDX #FE
```

```
     LDX #FF
```

Strona 62, linia 5
Brak znaku przeniesienia wyrazu do nowej linii

```
trom X i Y identyczne możliwości przesyłu, jak   LDA i STA aku
mulatorowi. Rozpatrzmy następujący program (dane dec):
```

```
trom X i Y identyczne możliwości przesyłu, jak   LDA i STA aku-
mulatorowi. Rozpatrzmy następujący program (dane dec):
```

Strona 62, linia 23
Zła forma wyrazu

```
     Innymi słowy program ten doda kolejne liczby 1+2+3+ ...
+19+20 i w S umieścić sumę liczb całkowitych z tego przedzia-
```

```
     Innymi słowy program ten doda kolejne liczby 1+2+3+ ...
+19+20 i w S umieści sumę liczb całkowitych z tego przedzia-
```

Strona 63, ostatnia linia
Błędna wartość (Uzasadnienie: w 4 linii od spodu wartość 00000010 bin = 2 dec)

```
jest tam wartość 3. Możemy jednak wykonać:
```

```
jest tam wartość 2. Możemy jednak wykonać:
```

Strona 68, linia 26
Literówka

```
komputerami opartymi na 6502 znaczne różnice. Podstawą w prze-
stawianiu znaków cyfr i liter oraz znaków specjalnych jest na
```

```
komputerami opartymi na 6502 znaczne różnice. Podstawą w przed-
stawianiu znaków cyfr i liter oraz znaków specjalnych jest na
```

Strona 69, linia 12
Literówka

```
     Należy zawrócić uwagę na jeszcze jedną istotną dla prog-
```

```
     Należy zwrócić uwagę na jeszcze jedną istotną dla prog-
```

Strona 72, linia 6 (pod rysunkiem)
Niepotrzebna litera (lub próba przenoszenia wyrazu z następnej linii)

```
     Rozkazy te, zwłaszcza LDA i STA, należą do najczęściej s
stosowanych. Z ich pomocą można realizować, jak o tym była mo-
```

```
     Rozkazy te, zwłaszcza LDA i STA, należą do najczęściej
stosowanych. Z ich pomocą można realizować, jak o tym była mo-
```

Strona 73, linie 1 i 2
Brak nawiasów

```
G, a G wartości H, np.: Z=G:H=G:G=Z  Z jest tu zmienną pomoc-
niczą .
```

```
G, a G wartości H, np.: Z=G:H=G:G=Z (Z jest tu zmienną pomoc-
niczą).
```

Strona 74, linia 28
Brak znaku przeniesienia wyrazu do nowej linii

```
     Po zakończeniu podprogramu, w celu odtworzenia poprzed
nich stanów rejestrów A, X i Y należy wykonać procedurę odwro-
```

```
     Po zakończeniu podprogramu, w celu odtworzenia poprzed-
nich stanów rejestrów A, X i Y należy wykonać procedurę odwro-
```

Strona 75, linia 21

Literówka

```
dawania i odejmowania z przeniesieniem. Możliwość zastosowanie
BCD podwaja w rzeczywistości liczbę rozkazów arytmetycznych.
```

```
dawania i odejmowania z przeniesieniem. Możliwość zastosowania
BCD podwaja w rzeczywistości liczbę rozkazów arytmetycznych.
```

Strona 76, linia 9

Literówka

```
znaczeniem operacji logicznych na bitach odpowiednie intruk-
cje pojawiają się coraz szerzej w językach programowania.
```

```
znaczeniem operacji logicznych na bitach odpowiednie instruk-
cje pojawiają się coraz szerzej w językach programowania.
```

Strona 78, linia 4

Brak nawiasów

```
nowe. Jeżeli użyjemy sekwencji  liczby w hex :
```

```
nowe. Jeżeli użyjemy sekwencji (liczby w hex):
```

Strona 81, linia 12

Błędny mnemonik rozkazu

```
AND, OR, EOR. Do tej grupy rozkazów o najbardziej rozbudowa-
```

```
AND, ORA, EOR. Do tej grupy rozkazów o najbardziej rozbudowa-
```

Strona 82, linia 6

Zdublowany wyraz

```
     Gdy do programu wprowadzimy taką sekwencję, po dojściu do
do niej wyprowadzanie danych na ekran zostanie zatrzymane do
```

```
     Gdy do programu wprowadzimy taką sekwencję, po dojściu
do niej wyprowadzanie danych na ekran zostanie zatrzymane do
```

Strona 82, linie: 26, 30 i 33

Literówka (w linii26) oraz błąd rzeczowy (w linii 30 i 33). Znacznik Z jest ustawiany (Z=1) gdy wynikiem ostatniej operacji (tutaj BIT) było zero, a kasowany (Z=0) w przeciwnym przypadku.

```
zycjach pary bitów 1. Na przykład, w zapisie binarnym pary ba-
tów:

     10101010         Wartość w akumulatorze
BIT  01010101         Wartość w pamięci

skasuje znacznik Z. Natomiast para:

     10101010         Wartość w akumulatorze
BIT  01001101         Wartość w pamięci

ustawi znacznik Z, ponieważ na pozycji b3 oba bity są jedynko-
```

```
zycjach pary bitów 1. Na przykład, w zapisie binarnym pary baj-
tów:

     10101010         Wartość w akumulatorze
BIT  01010101         Wartość w pamięci

ustawi znacznik Z. Natomiast para:

     10101010         Wartość w akumulatorze
BIT  01001101         Wartość w pamięci

skasuje znacznik Z, ponieważ na pozycji b3 oba bity są jedynko-
```

Strona 83, linia 26

Literówka

```
Dlatego właśnie przed dodawaniem trzeba go skasować, a przed
odejmowaniem ustawić, by zapobiec błędowi, który może wywołać
"utajona" wartość odziedziczona kiedy  przez znacznik.
```

```
Dlatego właśnie przed dodawaniem trzeba go skasować, a przed
odejmowaniem ustawić, by zapobiec błędowi, który może wywołać
"utajona" wartość odziedziczona kiedys przez znacznik.
```

Strona 88, linia 7

Błąd w programie. aby to udowodnić, wystarczy podstawić za G i H taką samą liczbę i prześledzić krok po kroku program: po INX, G będzie większe od H o 1. Po CPX H (czyli SBC [G]–H, po podstawieniu za G=H+1 mamy [H+1]–H=+1) C będzie sygnalizowało "brak pożyczki" (czyli 1) i nastąpi skok. Pożyczka wystąpi w przypadku, gdy G (czyli H-1) zmniejszymy o jeden: SBC [G]–H => [H–1]–H=–1, co da pożądany efekt.

```
     LDX G
     INX
     CPX H
     BCS SKOCZ
```

```
     LDX G
     DEX
     CPX H
     BCS SKOCZ
```

Strona 94, linie 8-10

Brak nawiasów (w liniach 8 i 10) oraz myślnika (w linii 9) między polską i angielską nazwą trybu adresowania

```
12. pośredni strony zerowej postindeksowany Y -
     indirect zero page postindexed Y                    2
      LDA  Z ,Y                      LDA  15 ,Y
13. pośredni  indirect  - tylko JMP                      3
      JMP (Q)                        JMP  4000
```

```
12. pośredni strony zerowej postindeksowany Y -
     indirect zero page postindexed Y                    2
      LDA (Z),Y                      LDA (15),Y
13. pośredni - indirect  - tylko JMP                     3
      JMP (Q)                        JMP (4000)
```

Strona 94, linia 26

Nieadekwatne mnemoniki rozkazów w stosunku do przypisanych im w tekście funkcji (zła kolejność)

```
jest przeważnie akumulator. W przesłaniach między pamięcią a
rejestrami także określa się w samym kodzie operacji, skąd
(np. STA) lub dokąd (np. LDY) mają być przesłane dane.
```

```
jest przeważnie akumulator. W przesłaniach między pamięcią a
rejestrami także określa się w samym kodzie operacji, skąd
(np. LDY) lub dokąd (np. STA) mają być przesłane dane.
```

Strona 97, linia 21

Błędny adres bazowy (i idący z nim błąd w rachunkach)

```
wartość akumulatora zapisana będzie w komórce o adresie 1127
(1000+27).
```

```
wartość akumulatora zapisana będzie w komórce o adresie 1127
(1100+27).
```

Strona 97, linie 31 i 32

Błędna konwersja z hex na dec (w linii 31) oraz błędny adres (w linii 32)

```
     Efektem działania programu będzie to, że 240 bajtów po-
czynając od adresu 2000 hex zostanie skopiowanych na odcinek o
```

```
     Efektem działania programu będzie to, że 224 bajtów po-
czynając od adresu 2001 hex zostanie skopiowanych na odcinek o
```

Strona 99, linia 9

Błędny warunek (rozkaz BNE) opuszczenia pętli. Aby skopiować dwa wektory (4 bajty) pętla musi zostać wykonana 4-krotnie (także dla Y=0, a nie tylko dla Y={3,2,1}). Rozwiązaniem problemu jest kontrolowanie znacznika wyniku ujemnego (zamiast zerowego), który pozwala wykonać pętlę również dla Y=0.

```
     LDY #3
CYKL LDX 0A,Y
     STX D0,Y
     DEY
     BNE CYKL
```

```
     LDY #3
CYKL LDX 0A,Y
     STX D0,Y
     DEY
     BPL CYKL
```

Strona 100, linia 18

Literówka

```
sorów 8-bitowych, kt re rozporządzają takim sposobem adreso-
```

```
sorów 8-bitowych, które rozporządzają takim sposobem adreso-
```

Strona 101, linie 11 i 13

Literówka (w linii 11) i brak operatora przypisania pomiędzy rejestrem i wartością (w linii 13)

```
     Pary te można odczytać jako adresy (bajty przedstawione):
3000, 3040, 3080. A oto przykładowa sekwencja rozkazów:
     LDX #0       X 0
```

```
     Pary te można odczytać jako adresy (bajty przestawione):
3000, 3040, 3080. A oto przykładowa sekwencja rozkazów:
     LDX #0       X=0
```

Strona 104, linie: 5-10 (ciało programu)

Brak rozkazu sterującego licznikiem pętli (zmniejszającego wartość załadowaną na początku programu do rejestru Y)

```
     LDY #3F      Zapisujemy w Y długość bloków -1
CYKL CLC          Zawsze przed dodawaniem ...
     LDA (CC),Y   Liczba z pierwszego ciągu w A
     ADC (CE),Y   Dodanie liczby z drugiego ciągu
     STA (D0),Y   Zapisanie wyniku w trzecim ciągu
     BPL CYKL
```

```
     LDY #3F      Zapisujemy w Y długość bloków -1
CYKL CLC          Zawsze przed dodawaniem ...
     LDA (CC),Y   Liczba z pierwszego ciągu w A
     ADC (CE),Y   Dodanie liczby z drugiego ciągu
     STA (D0),Y   Zapisanie wyniku w trzecim ciągu
     DEY
     BPL CYKL
```

Strona 105, linia 2

Literówka

```
     Do zmodyfikowanej werdji omawianego w tej książce mikro-
```

```
     Do zmodyfikowanej wersji omawianego w tej książce mikro-
```

Strona 105, linia 24

Literówka

```
     Wprowadzono komunikację rejestrów X i Y zes stosem:
```

```
     Wprowadzono komunikację rejestrów X i Y ze stosem:
```

Strona 107, ostatnia linia

Literówka

```
sprawa przekazania do podprogramu danych wyjściowych, a z
```

```
sprawa przekazania do podprogramu danych wejściowych, a z
```

Strona 108, linia 10 (od spodu)

Zła forma wyrazu

```
     Warunkiem korzystania do tego celu z rejestrów jest ich
dostępność w danej chwili. Taki sposób przekazywania paramet-
```

```
     Warunkiem wykorzystania do tego celu rejestrów jest ich
dostępność w danej chwili. Taki sposób przekazywania paramet-
```

Strona 112, linie 5-14 (od spodu – przykład dzielenia binarnego)

Pominięte operacje arytmetyczne: dzielenia i odejmowania

```
  199                    11000111
34427:173              1000011001111011:10101101
173                    10101101
1712                     10111111
1557                     10101101
 1557                       100101110
 1557                        10101101
    0                         10101101
                              10101101
                                     0
```

```
  199                    11000111
34427:173              1000011001111011:10101101
173                    10101101
1712                     10111111
1557                     10101101
 1557                       100101110
 1557                        10101101
    0                       100000011
                             10101101
                               10101101
                               10101101
                                      0
```

Strona 113, linia 9

Błąd rzeczowy. Aby zmieścić wynik z dzielenia 16-bitowej liczby przez 8-bitową na 8-śmiu bitach należy dobrać 8-bitowy dzielnik, tak aby MSB(dzielnej)*256 / DZIELNIK < 256. Po obustronnym wymnożeniu nierówności przez dodatni DZIELNIK daje to nam MSB(dzielnej)*256 < DZIELNIK*256, co stanowi dowód i uzasadnienie twierdzenia.

```
szta w A. Wystarczającym warunkiem poprawnego działania algory-
tmu jest to, by dzielnik nie przewyższał bardziej znaczącego
bajtu dzielnej znajdującego się w akumulatorze.
```

```
szta w A. Wystarczającym warunkiem poprawnego działania algory-
tmu jest to, by dzielnik przewyższał wartość bardziej znaczącego
bajtu dzielnej znajdującego się w akumulatorze.
```

Strona 113, linie: 13-25 (ciało programu)

Brak warunku sprawdzającego czy po rozkazie ROL A (linia 17) z akumulatora do C nie został przeniesiony najbardziej znaczący bit o wartości 1, co oznaczałoby, że złożona z nich 9-bitowa liczba (z ustawionym najstarszym bitem >> tutaj znacznikiem C) jest większa od dzielnika i należałoby wykonać odejmowanie. Tymczasem wykonywane po tym rozkazie porównanie CMP DK nie bierze znacznika C pod uwagę ustawiając go tylko na podstawie liczby zawartej w akumulatorze i pamięci (tej pod adresem DK).
Możliwa zatem jest sytuacja (analizując program od pierwszej linii z wartościami:
A=11111110, DA=11111111, DK=11111111):

```
CMP DK      A<DK więc C=0
BCS POWR    pomijamy bo C<>1
LDX #8      nieistotne
ASL DA      DA=11111110,      C=1
ROL A       A =11111101,      C=0,     ...w tym miejscu z A wypada MSBit ...
CMP DK      ...a w tym ginie, bo 8-bitowy A<DK, zatem C=0
BCC OMIN    skok pod OMIN, gdy C=0, ... i odejmowanie jest pomijane.
```

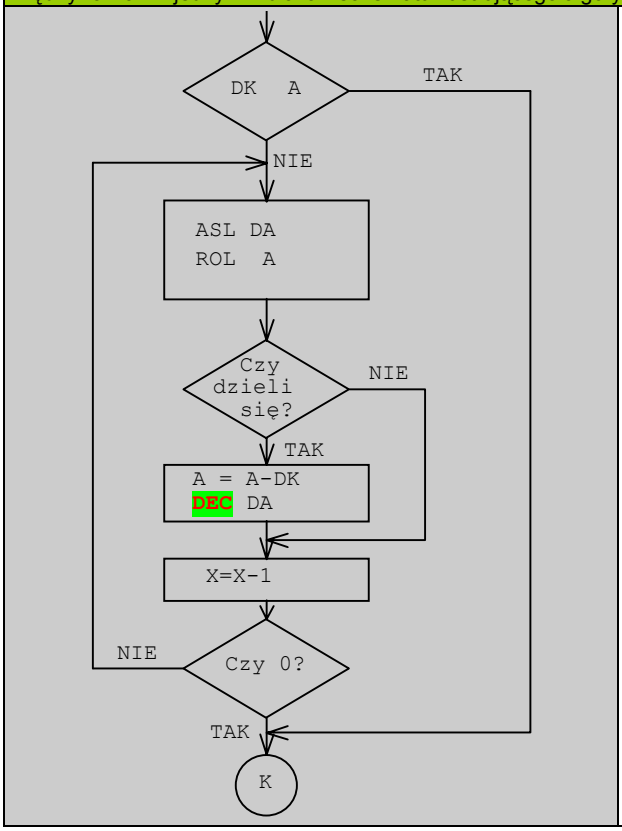
Strona 113, linie: 13-25 (ciało programu)

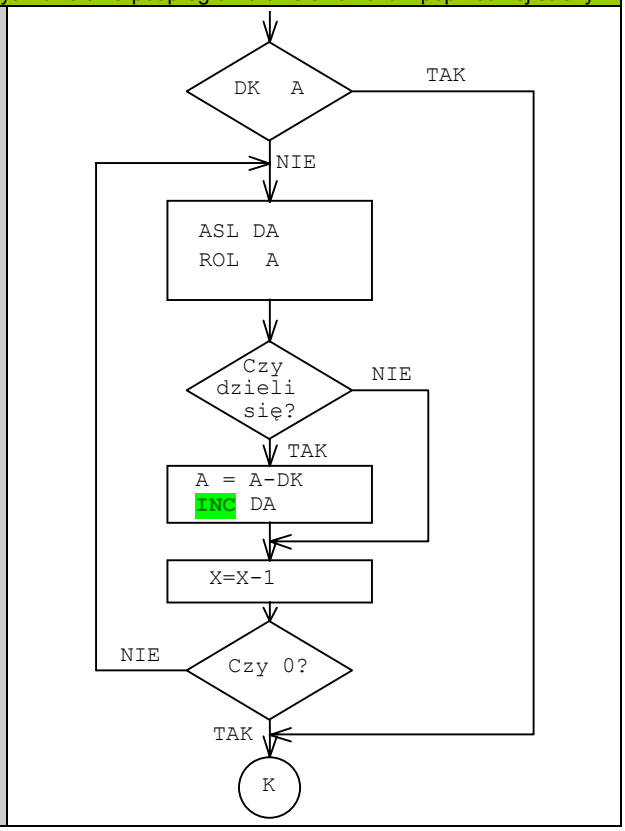
```
DZIEL CMP DK        Jeżeli A nie mniejsze od DK, to
      BCS POWR      skok na koniec programu
      LDX #8        Licznik przesunięć bitów
CYKL  ASL DA        Przesunięcie w LSB dzielnej
      ROL A         i przejęcie najwyższego bitu do A
      CMP DK        Porównanie z dzielnikiem
      BCC OMIN      Jeżeli A <DK, to pomija odejmowanie
      SBC DK        Odejmuje dzielnik od A, C już ustawiony
      INC DA        Wstawia 1 na koniec LSB dzielnej
OMIN  DEX           Licznik zmniejszony o 1
      BNE CYKL      Jeżeli nie zero, powtórzenie czynności
      STA RESZTA    Zapisuje resztę z dzielenia
POWR  RTS           Powrót z podprogramu.
```

```
DZIEL CMP DK        Jeżeli A nie mniejsze od DK, to
      BCS POWR      skok na koniec programu
      LDX #8        Licznik przesunięć bitów
CYKL  ASL DA        Przesunięcie w LSB dzielnej
      ROL A         i przejęcie najwyższego bitu do A
      BCS ODEJM     Odejmuj gdy wysunięty z A do C bit MSB to 1
      CMP DK        Porównanie z dzielnikiem
      BCC OMIN      Jeżeli A <DK, to pomija odejmowanie
ODEJM SBC DK        Odejmuje dzielnik od A, C już ustawiony
      INC DA        Wstawia 1 na koniec LSB dzielnej
OMIN  DEX           Licznik zmniejszony o 1
      BNE CYKL      Jeżeli nie zero, powtórzenie czynności
      STA RESZTA    Zapisuje resztę z dzielenia
POWR  RTS           Powrót z podprogramu.
```

Strona 114, schemat blokowy

Błędny rozkaz w jednym z bloków schematu ilustrującego algorytm działania podprogramu dzielenia liczb z poprzedniej strony.





Strona 115-116, ciało podprogramu DZIEL1

Oprócz błędu ze strony 113, na samym początku podprogramu DZIEL1 pobierany jest ze stosu adres powrotny do programu głównego (DANE), a nie jak zamierzał autor dzielnik i LSB dzielnej. Ponadto wykonując ostatni rozkaz w tymże podprogramie - powrotu (RTS), zamiast do linii następującej po JSR DZIEL1, następuje skok pod adres $0000, który jest bezpośrednio wcześniej odkładany na stos (LDA #0; PHA; PHA). Dodatkowy błąd kryje się w części tegoż podprogramu przed etykietą POWR, gdyż w tymże momencie jego wykonywania (rozkaz PHA) odkładany jest ostatni parametr (iloraz) dla programu głównego i powinien nastąpić do niego powrót. Zamiast tego odkładany jest adres $0000. Wszystkie te błędy eliminuje zapamiętanie adresu powrotnego bezpośrednio po wywołaniu podprogramu DZIEL1 w komórkach ADRH i ADRL oraz jego odtworzenie poprzez odłożenie na stos bezpośrednio przed powrotem za pomocą RTS do programu głównego.

```
DZIEL1 PLA        Pobiera ze stosu  d z i e l n i k
       STA DK     Zapisuje go w komórce o adresie DK
       PLA        Pobiera młodszy bajt dzielnej
       STA DA     Zapisuje go w komórce o adresie DA
       PLA        Pobiera MSB dzielnej i zostawia  w A
       CMP DK     Odtąd część obliczeniowa bez zmian
       BCS POWR
       LDX #8
CYKL ASL DA
     ROL A
     CMP DK
     BCC OMIN
     SBC DK
     INC DA
OMIN DEX
     BNE CYKL     Koniec obliczeń
     PHA          Przekazuje na stos  r e s z t ę
     LDA DA       Pod adresem DA jest teraz iloraz
     PHA          Przekazuje go na stos
POWR LDA #0       Wyjaśnienie w tekście
     PHA
     PHA
     RTS          Powrót do programu głównego.
```

```
DZIEL1 PLA; STA ADRL; PLA; STA ADRH   Zachowuje adres powrotny
       PLA        Pobiera ze stosu  d z i e l n i k
       STA DK     Zapisuje go w komórce o adresie DK
       PLA        Pobiera młodszy bajt dzielnej
       STA DA     Zapisuje go w komórce o adresie DA
       PLA        Pobiera MSB dzielnej i zostawia  w A
       CMP DK     Odtąd część obliczeniowa bez zmian
       BCS POWR
       LDX #8
CYKL ASL DA
     ROL A
     BCS ODJM
     CMP DK
     BCC OMIN
ODJM SBC DK
     INC DA
OMIN DEX
     BNE CYKL     Koniec obliczeń
     PHA          Przekazuje na stos  r e s z t ę
     LDA DA       Pod adresem DA jest teraz iloraz
     PHA          Przekazuje go na stos
     JMP ADRS
POWR LDA #0       Wyjaśnienie w tekście
     PHA
     PHA
ADRS LDA ADRH; PHA; LDA ADRL; PHA   Odtwarza adres powrotny
     RTS          Powrót do programu głównego.
```

Strona 117, linia 27

Niepotrzebna litera (lub próba przenoszenia wyrazu z następnej linii)

```
ich równości, dalsze sprawdzanie przestaje być potrzebne i o
od razu następuje przeskok do etykiety relacja. Jeżeli znacz-
```

```
ich równości, dalsze sprawdzanie przestaje być potrzebne i
od razu następuje przeskok do etykiety relacja. Jeżeli znacz-
```

Strona 120, linia 29

Literówka

```
su w stosunku do początku tablicy będzie dwukrotnie większe
nim indeks elementu. Przypuśćmy, że chcemy ustalić, czy ele-
```

```
su w stosunku do początku tablicy będzie dwukrotnie większe
niż indeks elementu. Przypuśćmy, że chcemy ustalić, czy ele-
```

Strona 121, linia 20

Brak operatora

```
asemblerze oznaczenie mniej znaczącego bajtu adresu  T oraz
bardziej znaczącego  >T.
```

```
asemblerze oznaczenie mniej znaczącego bajtu adresu <T oraz
bardziej znaczącego  >T.
```

Strona 123, linia 30

Literówka

```
dla niej dodatkowy bajt. Oba adresy umieścimy na stronie zey-
rowej w bezpiecznym na Atari obszarze CB-D1 hex. Oto podprog-
```

```
dla niej dodatkowy bajt. Oba adresy umieścimy na stronie ze-
rowej w bezpiecznym na Atari obszarze CB-D1 hex. Oto podprog-
```

Strona 123, linie 33 i 35 oraz Strona 124, linia 32 (ciało podprogramu MOVE)

Ten sam błąd, który autor popełnił w programie na stronie 115 zapominając o tym, że po wywołaniu podprogramu rozkazem JSR na stos jest odkładany adres powrotny i należy go z niego zdjąć (i zapisać w bezpiecznym miejscu) zanim zacznie się pobierać argumenty, a po ich pobraniu i przed powrotem do programu głównego przed rozkazem RTS należy odłożyć spowrotem na stos zachowany adres powrotny. Ponadto w linii 35 na str.123 (BNE POWR) użyto etykiety, której nie zdefiniowano w żadnym miejscu programu zamiast wykonać skok pod etykietę KONC.

```
MOVE PLA           C
     CMP #3        Czy 3 parametry?
     BNE POWR      Jeżeli nie - wyjście z programu
     PLA
                                    Strona 124, linia 32:
KONC RTS        Powrót do programu.
```

```
MOVE PLA; STA AL; PLA; STA AH   Zachowanie adresu powrotnego
     PLA           C
     CMP #3        Czy 3 parametry?
     BNE KONC      Jeżeli nie - wyjście z programu
     PLA
                                    Strona 124, linia 32:
KONC LDA AH; PHA; LDA AL; PHA; RTS        Powrót do programu.
```

Strona 125, linia 14

Powołanie się na etykietę nie występującą w podprogramie MOVE na str.123-124.

```
     BNE NEXT
```

```
     BNE CYKL
```

Strona 126, linia 25

Brak znaku # przed argumentem rozkazu w trybie natychmiastowym.

```
     CPX  40        Czy ostatni?
```

```
     CPX #40        Czy ostatni?
```

Strona 126, linia 30

Brak znaku # przed argumentem rozkazu w trybie natychmiastowym.

```
     LDX  100-40
```

```
     LDX #100-40
```

Strona 127, linia 9

Zła forma wyrazu

```
     Jedną z ciekawych metod programowania, w wielu wypadków
pozwalającą udoskonalić program, jest metoda zwana modyfika-
```

```
     Jedną z ciekawych metod programowania, w wielu wypadkach
pozwalającą udoskonalić program, jest metoda zwana modyfika-
```

Strona 127, linia 14

Brak nawiasów

```
     Można to zilustrować prostym programem, który zapiszemy
w kodzie maszynowym i asemblerze  dane w hex .
```

```
     Można to zilustrować prostym programem, który zapiszemy
w kodzie maszynowym i asemblerze (dane w hex).
```

Strona 127, linia 29 i 30

Błędnie wyliczone adresy oraz wartość argumentu dla ostatniego rozkazu (BNE CYKL)

```
0604 C8                              INY
0605 D0 FC                           BNE CYKL
```

```
0606 C8                              INY
0607 D0 FA                           BNE CYKL
```

Strona 128, linia 5

Etykieta umieszczona po stronie programu w JM

```
0600 AD FE 20 LELA              LDA ARRAY
```

```
0600 AD FE 20             LELA  LDA ARRAY
```

Strona 128, linia 9

Brak kodu rozkazu RTS po stronie programu w JM

```
060B                      LELA1 RTS
```

```
060B 60                   LELA1 RTS
```

Strona 129, ostatnia linia

Błędna etykieta

```
ETYKIETY .BYTE L1-MOD1, L2-MOD2, L3-MOD1, L4-MOD1, L5-MOD1
```

```
ETYKIETY .BYTE L1-MOD1, L2-MOD1, L3-MOD1, L4-MOD1, L5-MOD1
```

Strona 135, linia 11

Zła forma wyrazu

```
skompilowane. Nie mogą one w pełni osiągnąć walorów programów
napisanych w JM, stanowią jednak dogodny pomost między obu
językami.
```

```
skompilowane. Nie mogą one w pełni osiągnąć walorów programów
napisanych w JM, stanowią jednak dogodny pomost między oby-
dwoma językami.
```

Strona 135, linie 17 i 18

Zła forma wyrazu w linii 17 oraz literówki w linii 18

```
ki ułatwiające  korzystanie z wstawek maszynowych. Obok pow-
szechnie stosowanych instrukcji PEEK i POKE do środków tych
zaliczyć trzeba użyteczne, a czasem niedostępną w innych wer-
sjach Basicu intrukcje służące do operawonia ciągami znakowy-
```

```
ki ułatwiające  korzystanie z wstawek maszynowych. Obok pow-
szechnie stosowanych instrukcji PEEK i POKE do środków tych
zaliczyć trzeba użyteczne, a czasem niedostępne w innych wer-
sjach Basicu instrukcje służące do operowania ciągami znakowy-
```

Strona 135, linia 29

Literówka

```
s t o s. Dane niezbędne do wykonanie podprogramu przekazuje
```

```
s t o s. Dane niezbędne do wykonania podprogramu przekazuje
```

Strona 136, linie 33 i 34

Niezgodna z rzeczywistością kolejność ułożenia bajtów adresu powrotnego na stosie

```
     Szczyt+5   xx      Starszy bajt adresu powrotu
     Szczyt+6   xx      Młodszy bajt adresu powrotu.
```

```
     Szczyt+5   xx      Młodszy bajt adresu powrotu
     Szczyt+6   xx      Starszy bajt adresu powrotu.
```

Strona 137, linia 11

Literówka

```
     Kontynuując nasz przykład rozpatrz y teraz strukturę pod-
```

```
     Kontynuując nasz przykład rozpatrzmy teraz strukturę pod-
```

Strona 138, linie 7 i 8

Błędy w programie: w linii nr 20 w poleceniu INPUT A,BB używa się zmiennej BB (zamiast B), która nie jest nigdzie wykorzystywana, natomiast w linii nr 30 zamiast znaku * (mnożenie) używa się znaku x.

```
     20 ?  "podaj dwie wartości": INPUT A,BB
     30 X=USR(1536,A,B):? PEEK(16000)+256xPEEK(16001):END
```

```
     20 ?  "podaj dwie wartości": INPUT A,B
     30 X=USR(1536,A,B):? PEEK(16000)+256*PEEK(16001):END
```

Strona 140, linie 2 i 3

Brak nawiasów

```
tać  po prawej  d z i e s i ę t n e  wartości kodu maszynowe-
go :
```

```
tać (po prawej  d z i e s i ę t n e  wartości kodu maszynowe-
go):
```

Strona 140, linie 15 i 17

Błędy w programie: próba przypisania wartości liczbowej zmiennej łańcuchowej A$ (2-wymiarowej?) bez przekonwertowania jej za pomocą CHR$ do kodu znaku i podstawieniu do kolejnej komórki zmiennej A$ (w linii nr 20) oraz brak rozkazu POKE 766,0 na końcu programu w linii nr 40 umożliwiającego zatwierdzenie generowanego przez program kodu BASICu.

```
     10 DIM A$(12)
     20 FOR I=1 TO 12: READ DANE: A$(I, I)= DANE: NEXT I
     30 DATA 104,104,104,133,0,104,104,37,0,133,0,96
     40 POKE 766,1: ? "20 A$="; CHR$(34); A$
```

```
     10 DIM A$(12)
     20 FOR I=1 TO 12: READ DANE: A$(I)=CHR$(DANE): NEXT I
     30 DATA 104,104,104,133,0,104,104,37,0,133,0,96
     40 POKE 766,1: ? "20 A$="; CHR$(34); A$: POKE 766,0
```

Strona 140, linie 21 i 22

Literówki

```
kursorem po tej linii, co wprowadzi ją do naszego programujja-
ko nową linię o numerze 20. Zamist poprzednich dopisujemy nowe
```

```
kursorem po tej linii, co wprowadzi ją do naszego programu ja-
ko nową linię o numerze 20. Zamiast poprzednich dopisujemy nowe
```

Strona 140, linia 25

Brak nawiasu otwierającego po instrukcji USR

```
     40 X=USR ADR(A$),U,W)
```

```
     40 X=USR(ADR(A$),U,W)
```

Strona 142, linia 5

Literówka

```
niego wszelkich skoków JMP i JSR. Jeżeli chodzi i JMP, to
```

```
niego wszelkich skoków JMP i JSR. Jeżeli chodzi o JMP, to
```

Strona 143, linia 12

Literówka

```
nie przybrał wartości kodu cydzysłowu lub Returnu.
```

```
nie przybrał wartości kodu cudzysłowu lub Returnu.
```

Strona 145, linia 17

Niepotrzebny znak dwukropka na końcu linii (powodujący błąd w programie)

```
40 FOR J=0 TO 7: D=PEEK(I+J): ? " ";:X=USR(A,D):NEXT J: ? " ";:
```

```
40 FOR J=0 TO 7: D=PEEK(I+J): ? " ";:X=USR(A,D):NEXT J: ? " ";
```

Strona 150, linia 35

Literówka

```
   Nie ma zatem innego sposobu rozmieszczenie w pamięci dwóch
```

```
   Nie ma zatem innego sposobu rozmieszczenia w pamięci dwóch
```

Strona 153, linia 5

Brak kropki

```
sterowanej licznikiem    Pętle tego rodzaju mają odrębne in-
```

```
sterowanej licznikiem.   Pętle tego rodzaju mają odrębne in-
```

Strona 153, linia 19

Odwołanie do niewłaściwego podpunktu

```
bądź omówionego w punkcie 8.6  indeksowania ujemnego można
```

```
bądź omówionego w punkcie 6.8  indeksowania ujemnego można
```

Strona 154, linia 19

Błąd w programie. Brak rozkazu inicjalizującego licznik pętli (rejestr X) na początku programu (czyli LDX), zamiast którego pojawił się rozkaz LDA #8. Bezsens zastosowania tego rozkazu widać w kolejnej linii (CYKL LDA P,X), w której następuje ponowne załadowanie wartości do akumulatora.

```
     LDA #8
CYKL LDA P,X
     STA Q,X
     DEX
     BNE CYKL
```

```
     LDX #8
CYKL LDA P,X
     STA Q,X
     DEX
     BNE CYKL
```

Strona 154, linia 31

Literówka

```
     Sprawdzenia warunku na początku jest w pełni możliwe, a-
```

```
     Sprawdzenie warunku na początku jest w pełni możliwe, a-
```

Strona 155, linia 17

Literówka

```
nie. Ponieważ STA nie wpływa na znaczniki, stan znacznika Z
zależałby od rozkazu LDA P,X czyli od sprawdzenia, jaką war-
tość została pobrana do akumulatora spod adresu Q+X. Gdyby by-
```

```
nie. Ponieważ STA nie wpływa na znaczniki, stan znacznika Z
zależałby od rozkazu LDA P,X czyli od sprawdzenia, jaka war-
tość została pobrana do akumulatora spod adresu Q+X. Gdyby by-
```

Strona 156, linia 26

Próba przenoszenia wyrazu z następnej linii

```
     Jednym ze sposobów realizacji CASE jest porównywanie zmie
zmiennej sterującej umieszczonej w akumulatorze z kolejnymi
```

```
     Jednym ze sposobów realizacji CASE jest porównywanie
zmiennej sterującej umieszczonej w akumulatorze z kolejnymi
```

Strona 157, linia 11

Błędny zakres

```
     4004-4006                    10 30
```

```
     4004-4005                    10 30
```

Strona 157, linie 28 i 31

Błąd w programie (w linii 28) i związany z nim błąd w tekście zamieszczonym pod tymże programem (w linii 31). Użyty w programie rozkaz LDA nie "zapewni wykonania właściwego skoku". Autorowi chodziło o rozkaz JSR, co widać z kontekstu (patrz np.: poprzednia linia kodu).

```
     STA TAB+2   MSB modyfikowanego operandu JSR
TAB  LDA FFFF    FFFF będzie zmodyfikowane na 3412

     Efektem tej sekwencji będzie czynność wykonywana przez
CASE. Raz jeszcze poznaliśmy przy tym zalety modyfikacji adre-
sów. Oczywiście, ostatni rozkaz LDA może być zastąpiony przez
```

```
     STA TAB+2   MSB modyfikowanego operandu JSR
TAB  JSR FFFF    FFFF będzie zmodyfikowane na 3412

     Efektem tej sekwencji będzie czynność wykonywana przez
CASE. Raz jeszcze poznaliśmy przy tym zalety modyfikacji adre-
sów. Oczywiście, ostatni rozkaz JSR może być zastąpiony przez
```

Strona 158, ostatnia linia

Odwołanie do niewłaściwego podpunktu

```
tym kątem spojrzeć na przykładowy  podprogram przedstawiony w
punkcie 7.5.
```

```
tym kątem spojrzeć na przykładowy  podprogram przedstawiony w
punkcie 7.6.
```

Strona 162, linia 1

Zdublowana linia: ostatnia linia z poprzedniej strony

Strona 162, linia 10

Nieprawidłowo przeniesiony wyraz do nowego wiersza

```
tamtym modelu dodatkowej pamięci do 64KB, a także na zastosowa
waniu nowej obudowy. Zachowując taką samą jak 800XL przestrzeń
```

```
tamtym modelu dodatkowej pamięci do 64KB, a także na zastoso-
waniu nowej obudowy. Zachowując taką samą jak 800XL przestrzeń
```

Strona 165, linie 1 i 2

Brak nawiasów

```
użyteczną nie tylko w grach grafikę graczy-pocisków  ang.
player-missile graphics - PMG , która umożliwia sprawną ani-
```

```
użyteczną nie tylko w grach grafikę graczy-pocisków (ang.
player-missile graphics - PMG), która umożliwia sprawną ani-
```

Strona 165, linia 7

Niepotrzebny wyraz w zdaniu

```
kształt trybów graficznych oznaczonych przez w Basicu nume-
```

```
kształt trybów graficznych oznaczonych w Basicu nume-
```

Strona 166, linia 5

Brak nawiasów

```
CONTROL. W rejestrze CH pod adresem 2FC  764 dec  pojawia się
```

```
CONTROL. W rejestrze CH pod adresem 2FC (764 dec) pojawia się
```

Strona 166, linie 11 i 13

Brak nawiasów kwadratowych (w linii 11) oraz błędnie podana częstotliwość (w linii 13)

```
się być informacja Chadwicka  7 . Według niej europejskie A-
tari pracują o 25 proc. szybciej niż amerykańskie:z cyklem
2216 MHz. Ponieważ cykl zegarowy 6502 wynosi dwukrotnie mniej,
```

```
się być informacja Chadwicka [7]. Według niej europejskie A-
tari pracują o 25 proc. szybciej niż amerykańskie:z cyklem
2,216 MHz. Ponieważ cykl zegarowy 6502 wynosi dwukrotnie mniej,
```

Strona 166, linia 19

Literówka

```
go iinne przerwania. Czy jednak mimo wszystko nie jest to dla
```

```
go inne przerwania. Czy jednak mimo wszystko nie jest to dla
```

Strona 166, linia 32

Literówka

```
wcześniej opisanych układów sca onych.
```

```
wcześniej opisanych układów scalonych.
```

Strona 168, linia 32

Brak nawiasów

```
pikselem  ang. picture element - pixel , a także niejednakową
```

```
pikselem (ang. picture element - pixel), a także niejednakową
```

Strona 169, linia 21

Literówka

```
Zapewnia to swobodę wyboru miejsca ma listę obrazu, jak i na
```

```
Zapewnia to swobodę wyboru miejsca na listę obrazu, jak i na
```

Strona 172, linia 15

Brak nawiasów

```
              nii w trybie graficznym 0  2 ANTIC-u :
```

```
              nii w trybie graficznym 0 (2 ANTIC-u):
```

Strona 175, linia 8

Brak zaimka w zdaniu

```
on wykonywany zamiast programu OS. By umiejętnie posługiwać
możliwościami OS w asemblerze i JM, trzeba poznawać rozmiesz-
```

```
on wykonywany zamiast programu OS. By umiejętnie posługiwać się
możliwościami OS w asemblerze i JM, trzeba poznawać rozmiesz-
```

Strona 175, linia 18

Brak nawiasów

```
     Monitor wykonuje po włączeniu komputera zimny start
 skok do podprogramu pod adresem E477 , podczas którego zero-
```

```
     Monitor wykonuje po włączeniu komputera zimny start
(skok do podprogramu pod adresem E477), podczas którego zero-
```

Strona 178, linia 13
Zła forma wyrazu

```
moglibyśmy wpisać nasz program obsługi. Natomiast duże znacze-
nie ma wektora    C-D (12-13 dec). Jeżeli wpiszemy tu adres
```

```
moglibyśmy wpisać nasz program obsługi. Natomiast duże znacze-
nie ma wektor     C-D (12-13 dec). Jeżeli wpiszemy tu adres
```

Strona 178, linia 24
Brak nawiasów

```
cią wyprowadzania obrazu na ekran. Są to przerwania: listy
obrazu  display list interrupt - DLI  i synchronizacji piono-
```

```
cią wyprowadzania obrazu na ekran. Są to przerwania: listy
obrazu (display list interrupt - DLI) i synchronizacji piono-
```

Strona 179, linia 24
Literówka

```
ścia. Adresy odpowiednich programów obługi znajdują się w
```

```
ścia. Adresy odpowiednich programów obsługi znajdują się w
```

Strona 180, linie 27 i 28
Literówki

```
     Posługiwanie się w przerwaniach zegarami POKEY-a możliwe
jest dzięki temu, że gdy licząc malejąco dochodzą do zera, u-
ruchamiamy jest program, którego adres startu zapisze się w
odpowiednim wektorze na stronia 2.
```

```
     Posługiwanie się w przerwaniach zegarami POKEY-a możliwe
jest dzięki temu, że gdy licząc malejąco dochodzą do zera, u-
ruchamiany jest program, którego adres startu zapisze się w
odpowiednim wektorze na stronie 2.
```

Strona 181, linia 9
Zła forma wyrazu

```
nienie grafiki graczy-pocisków, manewrowanie paru zbiorami
znaków itp.
```

```
nienie grafiki graczy-pocisków, manewrowanie paroma zbiorami
znaków itp.
```

Strona 182, linia 2
Zła forma wyrazu w komentarzu

```
     LDA #0           Wpisanie adres 600 do VDSLST
```

```
     LDA #0           Wpisanie adresu 600 do VDSLST
```

Strona 182, linia 13
Literówka w nazwie etykiety

```
VSYNC  = D40A         Rejestr ten - wait for horizontal syn-
```

```
WSYNC  = D40A         Rejestr ten - wait for horizontal syn-
```

Strona 182, linia 19
Brak symbolu # przed argumentem rozkazu w trybie natychmiastowym

```
     LDA  DE          Kolor żółto-zielony, duża jasność
```

```
     LDA #DE          Kolor żółto-zielony, duża jasność
```

Strona 182, linia 31
Literówka

```
w stosowanym w USA systemie NTCS wiązka elektronów wywołująca
```

```
w stosowanym w USA systemie NTSC wiązka elektronów wywołująca
```

Strona 185, linia 24

Błąd w programie: Procedura obsługi przerwania nie może kończyć się rozkazem JSR do podprogramu OS, który to kończy się rozkazem RTI, co spowodowałoby powrót z przerwania i pobranie ze stosu niewłaściwych danych (licznika programu PC i stanu rejestru znaczników P)

```
      JSR XITVBV
```

```
      JMP XITVBV
```

Strona 187, linia 2

Brak operatora

```
       LDA # REP
```

```
       LDA #<REP
```

Strona 188, linia 33

Literówka

```
      Obok wspomianych ośmiu IOCB w skład systemu wchodzą: po-
```

```
      Obok wspomnianych ośmiu IOCB w skład systemu wchodzą: po-
```

Strona 191, ostatnia linia

Brak przyimka w zdaniu

```
cjom, jakie wykonuje CIO. Przy każdej z nich ICBA powinien
znajdować się adres buforu dla wprowadzania lub wyprowadzania
```

```
cjom, jakie wykonuje CIO. Przy każdej z nich w ICBA powinien
znajdować się adres buforu dla wprowadzania lub wyprowadzania
```

Strona 193, linia 19

Niepotrzebnie zdublowana druga połowa wyrazu przeniesionego z końca poprzedniej linii

```
DBYTL/H   8-9     Wpisuje manipulator. Liczba przesłanych bajtów
                  tów
```

```
DBYTL/H   8-9     Wpisuje manipulator. Liczba przesłanych baj-
                  tów
```

Strona 194, linie: 16, 17 i 19

Literówka (w linii 16) i pourywane wyrazy (w liniach 17 i 19)

```
postaci skończonych ułamków dziesiętnych. W systemiem opera-
cyjnym Atari znajduje się pakiet programów, umożliwiających pos
posługiwanie się taką arytmetyką. Z pakietu tego korzysta A-
tari Basic. Możliwość posługiwania się ułamkami w sposób wz
```

```
postaci skończonych ułamków dziesiętnych. W systemie opera-
cyjnym Atari znajduje się pakiet programów, umożliwiających
posługiwanie się taką arytmetyką. Z pakietu tego korzysta A-
tari Basic. Możliwość posługiwania się ułamkami w sposób
```

Strona 195, linia 2

Brak nawiasów

```
szy niż 64  40 hex , wskazuje to na liczbę o wartości bezwzg-
```

```
szy niż 64 (40 hex), wskazuje to na liczbę o wartości bezwzg-
```

Strona 195, linia 22

Błędna nazwa rejestru OS

```
nie 5. Dwa 6-bajtowe pseudorejestry FR0 (adresy D4-D9) i  R1
```

```
nie 5. Dwa 6-bajtowe pseudorejestry FR0 (adresy D4-D9) i FR1
```

Strona 196, linia 31

Błędny adres podprogramu FST0P

```
FST0P DDA8   Zapisanie liczby FP z użyciem    FR0        70
             FLPTR
FST0P DDAB   Zapisanie liczby FP z użyciem    FR0        70
             FLPTR
```

Strona 196, linia 40
Błędny adres podprogramu AF1

```
AF1    DA46    Wyzerowanie rejestru w X        różne       80
AF1    DA48    Wyzerowanie rejestru w X        różne       80
```

Strona 197, linia 15
Błędny adres podprogramu FMOVE

```
FMOVE  = D0B6
```

```
FMOVE  = DDB6
```

Strona 199, linia 10
Literówka

```
       LDA # <LBUFF      Wpisuje fo INBUFF adres buforu
```

```
       LDA # <LBUFF      Wpisuje do INBUFF adres buforu
```

Strona 199, linia 20
Literówka

```
       .WORD START       o adresie uruchomienie programu
```

```
       .WORD START       o adresie uruchomienia programu
```

Strona 199, linia 32
Literówka

```
wet ten zwięzły przegląd pozwala lepiej uzmysłowi , jak trud-
```

```
wet ten zwięzły przegląd pozwala lepiej uzmysłowić, jak trud-
```

Strona 201, linia 27
Literówka

```
Dlatego wyposażenie się w nie jest pryktycznie konieczne.
```

```
Dlatego wyposażenie się w nie jest praktycznie konieczne.
```

Strona 203, linia 14
Błędny numer podrozdziału

```
10.1  Stosowanie makroasemblera MAC/65
```

```
10.2  Stosowanie makroasemblera MAC/65
```

Strona 203, linia 20
Brak znaku przeniesienia wyrazu do nowej linii

```
mocą można m.in. wygodnie tworzyć, a potem wykorzystywać bib
lioteki gotowych fragmentów programów. Przykładem takiej bib-
```

```
mocą można m.in. wygodnie tworzyć, a potem wykorzystywać bib-
lioteki gotowych fragmentów programów. Przykładem takiej bib-
```

Strona 203, linia 34
Literówka

```
jest przstrzeń pamięci edytora. MAC/65 jest kompatybilny "w
```

```
jest przestrzeń pamięci edytora. MAC/65 jest kompatybilny "w
```

Strona 204, linia 17
Literówka

```
li znaleźć pierwsze wystąpienie LDX, FIND =LDY=,A wskaże
```

```
li znaleźć pierwsze wystąpienie LDX, FIND =LDY=,A wskaże
```

Strona 204, linia 21
Literówka

```
     Istotnym u upełnieniem FIND jest REP (replace) pozwala-
```

```
     Istotnym uzupełnieniem FIND jest REP (replace) pozwala-
```

Strona 206, linia 35
Błędny operator

```
argument) korzystaliśmy: IRQ oznacza młodszy bajt adresu zaz-
```

```
argument) korzystaliśmy: <IRQ oznacza młodszy bajt adresu zaz-
```

Strona 207, linia 32
Błędnie skonwertowana wartość

```
110 .BYTE "ABC",3,-1

spowoduje wyprowadzenie w kodzie wynikowym wartości:

     41 42 43 02 FF
```

```
110 .BYTE "ABC",3,-1

spowoduje wyprowadzenie w kodzie wynikowym wartości:

     41 42 43 03 FF
```

Strona 208, linia 11
Błędnie skonwertowana wartość

```
200  .CBYTE 1,"SYSTEM"

wytworzy to ciąg: 01 53 59 53 54 45 CE. Kod znaku "M" został
zwiększony o 80 hex.
```

```
200  .CBYTE 1,"SYSTEM"

wytworzy to ciąg: 01 53 59 53 54 45 CD. Kod znaku "M" został
zwiększony o 80 hex.
```

Strona 210, linia 10
Literówka

```
jakie zapewnia MAC/65 w czasie asemblowanie.
```

```
jakie zapewnia MAC/65 w czasie asemblowania.
```

Strona 212, linia 5
Literówka

```
sób, że najprost ze są wykorzystywane we wnętrzu bardziej zło-
```

```
sób, że najprostsze są wykorzystywane we wnętrzu bardziej zło-
```

Strona 212, linia 11
Literówka

```
parą znaków %$ dła łańcuchów znakowych. Rozpatrzmy makrorozkaz
```

```
parą znaków %$ dla łańcuchów znakowych. Rozpatrzmy makrorozkaz
```

Strona 212, linia 14
Literówka

```
od 0 do 7, oznacazać będzie numer kanału, natomiast wartość
```

```
od 0 do 7, oznaczać będzie numer kanału, natomiast wartość
```

Strona 217, linia 3
Literówka

```
M sstart end dokąd
```

```
M  start end dokąd
```

Strona 217, linia 7
Literówka

```
bo też na oba te urządzenia (SP)listingów tworzony h przez
```

```
bo też na oba te urządzenia (SP)listingów tworzonych przez
```

Strona 217, linia 29
Literówka

```
sane. Jeżeli jej nie podamy, zapisamy będzie jeden sektor.
sane. Jeżeli jej nie podamy, zapisany będzie jeden sektor.
```

Strona 218, linia 12
Literówka

```
     Tak przygotowana wersja będzie doskonale wpółpracować z
     Tak przygotowana wersja będzie doskonale współpracować z
```

Strona 220, linia 4
Błędny numer podrozdziału

```
     1.1.1 - 3. Etykieta nie może być oznaczeniem kodu opera-
     1.1.3 - 3. Etykieta nie może być oznaczeniem kodu opera-
```

Strona 220, linia 7
Błędna odpowiedź do ćwiczenia

```
     1.5 - 3. d/ 110111011100000.
     1.5 - 3. d/ 1101110111000000.
```

Strona 220, linia 15
3 błędy: niepoprawne oznaczenie podpunktu b przez a, oznaczenie trybu adresowania natychmiastowego symbolem = zamiast # oraz nieoznaczenie ostatniej sekwencji rozkazów podpunktem e

```
     a/ LDA G3        d/ LDA =0              INC E4
     b/ LDA G3        d/ LDA #0            e/ INC E4
```

Strona 220, linia 18
Błędny kod w Basicu w podpunkcie c

```
     1.8 - 2. b/  U=62:W=62   c/ T=S:T+1:U=S
     1.8 - 2. b/  U=62:W=62   c/ T=S:T=T+1:U=S
```

Strona 220, linia 28
Brak odstępu pomiędzy mnemonikiem a operandem

```
     INCL
     INC L
```

Strona 221, linie 4 i 6
Błędnie przepisana wartość z zadania (linia 4) oraz wynikający z tego błędny wynik (w linii 6)

```
     110111          10110          100101
   - 100100        - 1011         -   111
     010011          1011           11110

     110111         100110          100101
   - 100100        -  1011         -   111
     010011         11011           11110
```

Strona 221, linia 25
2 błędy: brak odstępu pomiędzy mnemonikiem a operandem oraz zapisanie wyniku w niewłaściwej zmiennej

```
        INCQ                  STA W
        INC Q                 STA U
```

Strona 221, linia 27
Brak znaku przeniesienia wyrazu do nowej linii

```
hex. W drugim przykładzie ustawienie znacznika C przed dodawa
niem da poprawny wynik, gdy W+9 nie jest większe niż FF, nas-
hex. W drugim przykładzie ustawienie znacznika C przed dodawa-
niem da poprawny wynik, gdy W+9 nie jest większe niż FF, nas-
```

Strona 221, linia 34

Błędny numer ćwiczenia

```
     3.11 - 4. 90; 144. W kodzie BCD 4-krotne  ASL jest równo-
```

```
     3.11 - 1. 90; 144. W kodzie BCD 4-krotne  ASL jest równo-
```

Strona 222, linia 6

Błędny numer podrozdziału

```
     4.2.2 - 1. b/:
```

```
     4.4.2 - 1. b/:
```

Strona 222, linia 8

Błędnie skonwertowana na system binarny wartość C5 hex

```
     11110110
ORA  10100111
     11110111
```

```
     11110110
ORA  11000101
     11110111
```

Strona 222, linia 17

Brak rozwiązania ćwiczenia 2 z podrozdziału 4.6.

```
     4.6 - 2. Rozkaz BIT nie posiada trybu adresowania
natychmiastowego. Zamiast rozkazu BIT powinien być rozkaz AND.
```

Strona 222, linia 28

Brak rozkazu sterującego licznikiem pętli (zmniejszającego wartość załadowaną na początku programu do rejestru Y)

```
     LDY #3F
CYKL CLC
     LDA 3000,Y
     ADC 3040,Y
     STA 3080,Y
     BPL CYKL
```

```
     LDY #3F
CYKL CLC
     LDA 3000,Y
     ADC 3040,Y
     STA 3080,Y
     DEY
     BPL CYKL
```

Strona 223, linia 14

Błędny argument rozkazu STA

```
     STA T,K
```

```
     STA T,X
```

Strona 223, linia 26

Błędny numer podrozdziału

```
     7.5 - 2. Jedna z możliwości. Linie 60-80 należy skasować
```

```
     7.6 - 2. Jedna z możliwości. Linie 60-80 należy skasować
```

Strona 225, linia 16

Literówka

```
     15. Gilmore Ch.: Wwiedienfje w mikroprocessornuju tiech-
```

```
     15. Gilmore Ch.: Wwiedienije w mikroprocessornuju tiech-
```

Strona 228, linia 17

Nieprawidłowo przeniesiony wyraz do nowego wiersza

```
rozkazem odgałęzienia traktowane jako liczba  z e  z n a k i
k i e m, dodawana do owego adresu. Tak więc zasięg skoku waru-
```

```
rozkazem odgałęzienia traktowane jako liczba  z e  z n a -
k i e m, dodawana do owego adresu. Tak więc zasięg skoku waru-
```

Strona 229, linia 8

Brak nawiasów

```
nik wyniku zerowego Z jest ustawiony  Z=1 . Gdy Z=0, wykony-
```

```
nik wyniku zerowego Z jest ustawiony (Z=1). Gdy Z=0, wykony-
```

Strona 230, linia 5

Błąd rzeczowy

```
     Opis: Wykonuje czynności analogiczne jak BCC, gdy znacz-
nik wyniku ujemnego jest skasowany (Z=0). Gdy Z=1, wykonywany
jest następny rozkaz.
```

```
     Opis: Wykonuje czynności analogiczne jak BCC, gdy znacz-
nik wyniku zerowego jest skasowany (Z=0). Gdy Z=1, wykonywany
jest następny rozkaz.
```

Strona 231, linia 24

Brak znaku przeniesienia wyrazu do nowej linii

```
     Zastosowania: Niezbędny jest przed każdą operacją dodawa
nia. W przypadku dodawania liczb wielobajtowych należy zasto-
```

```
     Zastosowania: Niezbędny jest przed każdą operacją dodawa-
nia. W przypadku dodawania liczb wielobajtowych należy zasto-
```

Strona 233, ostatnia linia

Brak nawiasów

```
wynik w tym bajcie  M=M-1 . Wpływa na znaczniki N i Z.  N i e
```

```
wynik w tym bajcie (M=M-1). Wpływa na znaczniki N i Z.  N i e
```

Strona 235, linia 27

Literówka

```
pamięci. W celu zapewnienia powrotu z podprogramu adres nastę-
nego po JSR rozkazu  pomniejszony dla uproszczenia pracy 6502
```

```
pamięci. W celu zapewnienia powrotu z podprogramu adres nastę-
pnego po JSR rozkazu pomniejszony dla uproszczenia pracy 6502
```

Strona 239, linia 25

Brak nawiasów

```
     Opis: Zapewnia powrót z podprogramu. Odtwarza ze stosu
 zwiększając o 1  przechowany automatycznie przez JSR licznik
```

```
     Opis: Zapewnia powrót z podprogramu. Odtwarza ze stosu
(zwiększając o 1) przechowany automatycznie przez JSR licznik
```

Strona 240, linia 12

Literówka

```
znaku. N=1 oznacza, że bit znaku w wyniki przybrał wartość 1.
```

```
znaku. N=1 oznacza, że bit znaku w wyniku przybrał wartość 1.
```

Strona 241, linia 15

Wymieniony rozkaz CMP nie zostawia wyniku w akumulatorze

```
czeń (po ADC, SBC, CMP i wielu innych rozkazach), STA służy
```

```
czeń (po ADC, SBC, AND i wielu innych rozkazach), STA służy
```

Strona 242, linia 4

Literówka

```
rejstrze X, a późniejsze TXA odtworzyć ją w A. Gdy X wykorzy-
```

```
rejestrze X, a późniejsze TXA odtworzyć ją w A. Gdy X wykorzy-
```

Strona 242, linia 7

Mnemonik nieadekwatny do opisu

```
TYA       transfer accumulator into Y          Znaczniki: NZ
```

```
TAY       transfer accumulator into Y          Znaczniki: NZ
```

Strona 243, tabela, kolumna 1

Błędna długość rozkazu

```
ASL Q       0E    14   2    6
```

```
ASL Q       0E    14   3    6
```

Strona 243, tabela, kolumna 2

Błędna wartość kodu rozkazu

```
DEC Q       CE   216   3    6
```

```
DEC Q       CE   206   3    6
```

Strona 243, tabela, kolumna 2

Błędny czas wykonania (liczba cykli według Atari800Win PLus 3.1)

```
DEC Z,X     D6   214   2    5
```

```
DEC Z,X     D6   214   2    6
```

Strona 244, tabela, kolumna 2

Brak informacji o ilości cykli

```
TXA         8A   138   1
```

```
TXA         8A   138   1    2
```

Strona 246, tabela, kolumna 1

Niedozwolony rejestr indeksowy w operandzie rozkazu LDX

```
B6  182  LDX Z,X
```

```
B6  182  LDX Z,Y
```

Strona 247, tabela A5, kolumna "4 cyfra", wiersz "5"

Błędna wartość

```
1338
```

```
1388
```

Strona 247, tabela A5, kolumna "5 cyfra", wiersz "9"

Błędna wartość

```
16F90
```

```
15F90
```

Strona 248, podkreślony podpis nad listą rozkazów (i trybami adresowania) z grupy m2

Nieadekwatna grupa kodów trybów adresowania

```
    | 1 | m2 | a0 | 0 0 |

          Wartości a1

  m2 = 0  STY 1,3,5
       1  LDY 0,1,3,5,7
       2  CPY 0,1,3
       3  CPX 0,1,3
```

```
    | 1 | m2 | a0 | 0 0 |

          Wartości a0

  m2 = 0  STY 1,3,5
       1  LDY 0,1,3,5,7
       2  CPY 0,1,3
       3  CPX 0,1,3
```

Strona 248

Brak listy rozkazów z grupy m4

```
m4 = 0   ORA 1÷7
     1   AND 1÷7
     2   EOR 1÷7
     3   ADC 1÷7
     4   STA 0,1,3÷7
     5   LDA 1÷7
     6   CMP 1÷7
     7   SBC 1÷7
```

Strona 250, linie: 8-10

Błędne kody znaków w hex

```
7C   125   Oczyść ekran
7D   126   Usuń wstecz
7E   127   Tabulator
```

```
7D   125   Oczyść ekran
7E   126   Usuń wstecz
7F   127   Tabulator
```

Strona 251, punkt 1.7

Literówka

```
   1.7  Jak zapisac program w asemblerze?          19
```

```
   1.7  Jak zapisać program w asemblerze?          19
```

Strona 251, punkt 3.10

Literówka

```
   3.10 Przesunięcie i obrot bitów                 63
```

```
   3.10 Przesunięcie i obrót bitów                 63
```

Strona 252, punkt 3.13

Literówka

```
   3.13 Wnioski z tresci rozdziału                  69
```

```
   3.13 Wnioski z treści rozdziału                  69
```

Strona 252, linia 4

Literówka

```
Rozdział 4: Pięćdziesiąt sześc rozkazów             71
```

```
Rozdział 4: Pięćdziesiąt sześć rozkazów             71
```

Strona 253, punkt 9.7.2

Niepotrzebna kropka po numerze rozdziału

```
       9.7.1 Rodzaje przerwań w Atari               177
       9.7.2. Przerwania w tworzeniu obrazu         181
       9.7.3 Przykład: przerwania klawiatury        186
```

```
       9.7.1 Rodzaje przerwań w Atari               177
       9.7.2 Przerwania w tworzeniu obrazu          181
       9.7.3 Przykład: przerwania klawiatury        186
```

Wykaz błędów (pozostawionych):

Strona 105, linie: 32, 33 i 35

Wymienione rozkazy (INC i DEC) w trybie absolutnym występowały w podstawowej wersji 6502 (patrz str. 96), natomiast wykazanemu w linii 35 rozkazowi JMP, który na 65C02 istnieje w nowym trybie adresowania (pośrednim strony zerowej nie indeksowanym) przypisano kod 6C hex, który jest kodem rozkazu JMP w trybie pośrednim.

```
     INC w absolutnym (1A);
     DEC w absolutnym (3A).
     Wprowadzono nowy tryb pośredni strony zerowej  nie indek-
sowany . Dostępne są w nim: LDA (B2), STA (92), JMP (6C),
```