[ARKO]
projekt MIPS — sprawozdanie

Andrzej Dackiewicz

20 maja 2014

1 Opis projektu

Projekt "Żółwiowa Grafika" umożliwia zapis w pliku o rozszerzeniu BMP ruchów "żółwia". Możliwe są następujące polecenia:

- *ustaw* — ustawia żółwia w odpowiednim miejscu
- *podnies* — podnosi żółwia i uniemozliwia zamalowywanie obrazka
- *opusc* — opuszcza żółwia i umozliwia zamalowywanie obrazka
- *naprzod* — powoduje, że żółw porusza się do przodu o podaną odległość
- *obrot* — obraca żółwia o podany kąt

2 Założenia i realizacja projektu

Aby program prawidlowo działał to należy korzystać z następujących plików:

- *pusty.bmp* — Plik .bmp do ktorego zapisywany jest wynik działania programu. Jego poczatkowa zawartosc jest na poczatku wyczyszczona (zamalowanie na bialo).
- *katy.txt* — Plik zawiera rozpiskę sinusów i cosinusów rożnych kątów (Jest to potrzebne do obliczania położenia końcowego żółwia). Są tam kąty od 0 do 89. Większa ilość kątów nie jest konieczna gdyż sinusy i cosinusy są obliczane w prawidlowy sposob (dopisując - przed sin bądź cos zależnie od kąta. Do zapisu większych kątów wykorzystuje się kąty do 90 stopni. Sin i cos w programie będzie "przyjmowal" wartosci od 0 do 10000 gdyz potem przy liczeniu położenia końcowego żółwia następuje dzielenie przez 10000. Zostalo to w taki sposób zrobione ze względu na prostotę wczytywania takich liczb z pliku.
- *turtle.txt* — Plik zawiera polecenia dla żółwia i argumenty z jakimi działać ma program.
- *projektzolw.asm* — Plik .asm zawiera kod źródłowy programu.

W rysowaniu odcinków wykorzystywany jest algorytm Bresenham'a
Najważniejsze etykiety programu:

- *tworziwykonaj* — Tutaj i w dalszych częściach odbywa się analiza wczytanych znaków z pliku turtle.txt i na podstawie tego wybierana jest akcja programu. Jeśli nie będzie operacji o wczytanej nazwie to program wyrzuci napis "Nie ma takiej instrukcji" i będzie kontynuował wykonywanie operacji od następnej (Jeśli taka jest) Jeśli nie będzie juz żadnych operacji to program zakończy swoje działanie

- *naprzod* — Tutaj jest wykorzystywany algorytm Bresenham'a i zapisywany jest odcinek w pliku pusty.bmp

- *czytajliczbe* — Tutaj odczytywana jest liczba występująca po poleceniu w pliku turtle.txt potrzebna do wykonania danej operacji gdywczytane zostaną już wszystkie liczby to plik turtle zostaje "przewinięty" do następnego wersu z poleceniem.

3 Przyklad dzialania projektu

Wyniki działania programu są zamieszczone w oddzielnym pliku: wyniki.pdf Jeśli w pliku turtle.txt znajdą się następujące komendy:

ustaw 30 30 30
naprzod 30

To na poczatku żółw zostanie ustawiony w punkcie 30,30 i pod kątem 30 stopni do pionu. Operacja naprzod spowoduje narysowanie kreski o dlugosci 30. Operacja podnies spowoduje, że w kolejnym poleceniu naprzod 30 zamiast rysowac kolejny odcinek tym razem nastąpi jedynie przemieszczenie żółwia (bez rysowania). Operacja opusc spowoduje opuszczenie żółwia i w kolejnym naprzod 30 bedzie możliwe juz rysowanie odcinka.

Wynikiem takich operacji będzie pierwszy rysunek.

Dalej jeśli dopisze się kilka komend i plik turtle.txt będzie zawieral taki tekst:

ustaw 30 30 30
naprzod 30
podnies
naprzod 30
opusc
naprzod 30

Dodatkowo do tego co byloby wcześniej wykonane operacja podnies spowoduje, że w kolejnym poleceniu naprzod 30 zamiast rysowac kolejny odcinek tym razem nastąpi jedynie przemieszczenie żółwia (bez rysowania). Operacja opusc spowoduje opuszczenie żółwia i w kolejnym naprzod 30 bedzie możliwe juz rysowanie odcinka.

Wynikiem takich operacji będzie drugi rysunek.

Gdy w pliku turtle.txt znajdzie się następujący tekst:

ustaw 30 30 30
naprzod 30
podnies
naprzod 30
opusc
naprzod 30
obrot 120
naprzod 30

Dodatkowo zostanie wykonany obrót o 120 stopni (w prawo) przez co kolejna operacja naprzod nie będzie rysowała juz odcinka na tej samej prostej co pozostale.

Wynikiem takich operacji będzie trzeci rysunek.

Gdy w do pliku turtle.txt wpiszemy:

ustaw 30 30 30
naprzod 30
podnies
naprzod 30
opusc
naprzod 30
obrot 120
naprzod 30
podnies
naprzod 30
opusc
naprzod 30

Dorysowana zostanie wtedy kolejny odcinek o dlugosci 30.

Wynikiem takich operacji będzie czwarty rysunek.

Na koniec gdy w pliku turtle.txt znajdzie się:

ustaw 30 30 30
naprzod 30
podnies
naprzod 30
opusc
naprzod 30
obrot 120
naprzod 30
podnies
naprzod 30
opusc
naprzod 30
obrot 120
naprzod 30
podnies
naprzod 30
opusc
naprzod 30

To nastąpi dodatkowy obrot o 120 stopni zgodnie z kierunkiem ruchu zegara i dorysowane zostaną kolejne odcinki.

Wynikiem ostatecznym takiego programu będzie piąty rysunek.

4 Kod źródłowy projektu

```
#
# PROJEKT MIPS
# "ZOLWIOWA GRAFIKA"
```

```
# WYKONAL: Andrzej Dackiewicz
#_____________________________________________________________________________________

.data
nazwa: .asciiz "pusty.bmp"
miej: .space 54
tab1: .space 576000#307200
tabela: .space 10000
tabela2: .space 100000
tablica: .space 1000
turtle: .asciiz "turtle.txt"
katy: .asciiz "katy.txt"
.macro cout (%x)
.data
etyk: .asciiz %x
.text
li $v0, 4
la $a0, etyk
syscall
.end_macro
.text

main:

li $v0, 13
la $a0, turtle
     li $a1, 0
     li $a2, 0
     syscall#otworzylem plik do zapisu
     blez $v0, zle
move $s5, $v0# s5 - file descriptor

     li $v0, 14
move $a0, $s5 # $s5 to file descriptor pliku turtle.txt
la $a1, tabela
li $a2, 10000
syscall#skopiowalem gotowy naglowek

li $v0, 16
move $a0, $s5
syscall# zamknij plik



# tutaj juz jest wczytany plik turtle.txt

li $v0, 13
la $a0, katy
     li $a1, 0
     li $a2, 0
```

```
     syscall#otworzylem plik do zapisu
     blez $v0, zle
move $s6, $v0# s6 - file descriptor

     li $v0, 14
move $a0, $s6 # $s6 to descriptor pliku katy.txt
la $a1, tabela2
li $a2, 100000
syscall#skopiowalem gotowy naglowek

li $v0, 16
move $a0, $s6
syscall# zamknij plik

#

li $v0, 13
la $a0, nazwa
     li $a1, 0
     li $a2, 0
     syscall#otworzylem plik do odczytu

     move $t9, $v0
     li $v0, 14
move $a0, $t9
la $a1, miej
li $a2, 54
syscall#skopiowalem gotowy naglowek

move $a0, $t9
li $v0, 16
syscall#zamknalem plik

#

li $v0, 13
la $a0, nazwa
     li $a1, 1
     li $a2, 0
     syscall#otworzylem plik do zapisu

     move $s7, $v0
     li $v0, 15
move $a0, $s7 # $s7 to descriptor pliku pusty.bmp
la $a1, miej
li $a2, 54
syscall#zapisalem naglowek

move $a0, $s7
```

```
li $s0, 19200 # ustawienie koloru na bialy
la $s1, tab1
li $s2, 255 #niebieski
li $s3, 255 #zielony
li $s4, 255 #czerwony

zapelnienie: # zamalowuje kazdy piksel pliku pusty.bmp na bialo ( czyszczenie pliku pusty.
sb $s2, ($s1)
add $s1, $s1, 1
sb $s3, ($s1)
add $s1, $s1, 1
sb $s4 ($s1)
add $s1, $s1, 1
sub $s0, $s0, 1
beqz $s0, czytajpolecenia  # koniec
b zapelnienie

czytajpolecenia: # tutaj zaczyna sie dzialanie programu na podstawie tego co zostalo odczy

b tworziwykonaj

zle:  # wyswietlenie komunikatu na wypadek nieznalezienia pliku
cout("nie otworzyl pliku")
li $v0, 10
syscall
b koniec




# t0 i t1 moga posluzyc do zachowania sin i cos


ustawdokatow: # ustawia $t0 na poczatek znakow z pliku katy.txt
# i przechodzi na kolejne linie az dojdzie do odpowiedniego sin i cos

lb $t0, ($s6)

beqz $t1, powrot
add $t1, $t1, -1
b przewinkaty

przewinkaty: # przewijanie az do nastepnej linii

#cout ("\nprzewijam katy !!!!\n") # $t9 wykorzystywane do skakania po znakach
lb $t9, ($s6)
beq $t9, $zero koniec
beq $t9, 13, czykonieckata # sprawdza czy po nastapil potem znak konca linii
add $s6, $s6, 1
```

```
b przewinkaty

czykonieckata: # jesli kolejny znak ma kod ascii 10 to oznacza ze koniec linii
# zmieniane jest $s6 w ktorym zapisany jest poczatek analizowago wiersza z katami
# ( $s6 jest przewijane na kolejna linie )

add $s6, $s6, 1
lb $t9, ($s6)
beq $t9, 10, nowes6kata
b przewin

nowes6kata: # zmiana $s6 ( na kolejny wiersz )
add $s6, $s6, 1
b ustawdokatow


pobierajsincos: # ustawienie $t0 i $t1 tak by mozna w nich przechowywac
# wczytywane sin i cos danego kata
# przejscie do odczytywania ( czytajsin )
li $t0, 0
li $t1, 0
b czytajsin

czytajsin: # czytanie zaczyna sie od czytania sinusa potem cosinusa
# $t8 sluzy do skakania po znakach
# na poczatek ustawione na poczatek danego wiersza ( $s6 )
lb $t8, ($s6) # $t0 sluzy do zachowania sinusa
beq $t8, ' ', czytajcos # przejscie do czytania cosinusa
blez $t8, powrot
blt $t8, '0' powrot
bgt $t8, '9', powrot
addiu $t8, $t8, -48 # odejmowanie 48 by $t8 nie mialo wartosci ascii a liczbe od 0 do 9
mul $t0, $t0, 10 # mnozenie dotychczasowej liczby razy 10
add $t0, $t0, $t8 # dodanie czesci liczby ( $t8 )
add $s6, $s6, 1 # przejscie na kolejny znak
b czytajsin


czytajcos: # czytanie cosinusa
# tutaj tak samo $t8 sluzy do skakania po znakach
# przejscie na kolejny znak
# poniewaz wczesniej $s6 wskazywalo na ostatni znak poprzedniej linii
add $s6, $s6, 1
# teraz nowa linia
lb $t8, ($s6) # $t1 sluzy do zachowania cosinusa
blez $t8, powrot
blt $t8, '0' powrot
bgt $t8, '9', powrot
addiu $t8, $t8, -48 # odjecie 48 by $t8 nie mialo wartosci ascii a liczbe od 0 do 9
mul $t1, $t1, 10 # mnozenie dotychczasowego cosinusa * 10 ( $t1 )
```

```
add $t1, $t1, $t8 # dodanie czesci cosinusa do $t1
b czytajcos



zamaluj: # współrz poziom t2, współrz. pionowa t3
# zamalowanie odpowiedniego piksela (x1,x2)
# $t2 - x1
# $t3 - y1
la $t8, tab1#tablica # ustawienie $t8 na poczatek pliku i obliczenie ktory piksel trzeba z
mul $t3, $t3, 480
add $t8, $t8, $t3
div $t3, $t3, 480
mul $t2, $t2, 3
add $t8, $t8, $t2
div $t2, $t2, 3

li $t7, 0 # zamalowanie na czarno wybranego piksela ( $t8 )

sb $t7, ($t8)
sb $t7, 1($t8)
sb $t7, 2($t8)

li $t8, 0

jr $ra

# pomocnicze etykiety do obliczania wspolczynnikow kx, ky, dx, dy
# sluzace do obslugi algorytmu Bresenham'a

ustawkx1: # ustawienie kx
li $s5, 1
b krok2
ustawkxn1:
li $s5, -1
b krok2

ustawky1: # ustawienie ky
li $s6, 1
b krok3

ustawkyn1:
li $s6, -1
b krok3

ustawdxnormalnie: # ustawienie dx
sub $t5, $t0, $t2
b krok4

ustawdxodwrotnie:
```

```
sub $t5, $t2, $t0
b krok4

ustawdynormalnie: # ustawienie dy
sub $t6, $t1, $t3
b krok5

ustawdyodwrotnie:
sub $t6, $t3, $t1
b krok5


# odczytywanie sin i cos z uwzglednieniem jaki jest to kat

przypadek1: # przypadek z katem powyzej 360 stopni


sub $t1, $t1, 360
jal ustawdokatow
jal pobierajsincos
b ustawilem

przypadek2: # przypadek z katem od 270 do 360 stopni

sub $t1, $t1, 270
jal ustawdokatow
jal pobierajsincos

mul $t1, $t1, -1

move $t9, $t0
move $t0, $t1
move $t1, $t9

b ustawilem

przypadek3: # przypadek z katem od 180 do 270 stopni
sub $t1, $t1, 180
jal ustawdokatow
jal pobierajsincos

mul $t0, $t0, -1
mul $t1, $t1, -1

b ustawilem

przypadek4: # przypadek z katem od 90 do 180 stopni
sub $t1, $t1, 90

cout ("c")
```

```
jal ustawdokatow
cout ("d")
jal pobierajsincos
cout ("e")
mul $t0, $t0, -1

move $t9, $t0
move $t0, $t1
move $t1, $t9

b ustawilem




ustawplusminus: # wczytuje sin i cos i ustala ich znaki
# rozpatruje sytuacje zaleznie od kata ( $t4 )
move $t1, $t4 # $t4 zawiera kat aktualny
bge $t4, 360, przypadek1
bge $t4, 270, przypadek2
bge $t4, 180, przypadek3
bge $t4, 90, przypadek4
# jesli zaden z tych przypadkow to znaczy ze kat od 0 do 90 stopni czyli
# nic nie trzeba zmieniac

jal ustawdokatow # przeskakuje po wierszach z katami
jal pobierajsincos # pobieranie sinusa i cosinusa
b ustawilem




naprzod: # operacja naprzod wykorzystuje algorytm Bresenhama
# do rysowania odcinkow

la $s6, tabela2 # ustawiam odpowiednio s6 by pokazywalo poczatek pliku z katami

jal ustawplusminus # czytanie sin i cos
ustawilem: # tutaj juz sin i cos sa znane

cout ("\nsin wynosi:\n")
move $a0, $t0
li $v0, 1
syscall


cout ("\ncos wynosi:\n")
move $a0, $t1
```

```
li $v0, 1
syscall

# obliczanie punktow koncowych i zapisanie w t0 t1
mul $t0, $t0, $t6
mul $t1, $t1, $t6
div $t0, $t0, 10000
div $t1, $t1, 10000
add $t0, $t0, $t2
add $t1, $t1, $t3
# parametry koncowe
cout ("\nx2 wynosi\n")
move $a0, $t0
li $v0, 1
syscall

cout ("\ny2 wynosi\n")
move $a0, $t1
li $v0, 1
syscall


beq $s0, 0, krok1 # rozpoczecie algorytmu Brensenhama

# jesli podniesiony to nic nie rysuje
# tylko zmiana polozenia
move $t2, $t0
move $t3, $t1
b przewin  # przewiniecie do nastepnej operacji

# Do tej pory poprawnie ustawiony poczatek i koniec kreski

# t2 - x1
# t3 - y1
# t0 - x2
# t1 - y2

#_____________________________________________________________________________________

# ALGORYTM BRESENHAM'A

krok1: # Ustawia kx

ble $t2, $t0, ustawkx1
b ustawkxn1

krok2: # Ustawia ky

ble $t3, $t1, ustawky1
b ustawkyn1
```

```
krok3: # Ustawia dx
# t5 - dx


bge $t0, $t2, ustawdxnormalnie
b ustawdxodwrotnie

krok4: # Ustawia dy
# t6 - dy

bge $t1, $t3, ustawdynormalnie
b ustawdyodwrotnie

krok5: # zamalowuje pierwszy piksel

jal zamaluj

krok6:
blt $t5, $t6, krok16

krok7:
move $t9, $t5
div $t9, $t9, 2

move $t0, $t5
krok8:

bgtz $t0, krok9_14
b przewin

krok9_14:
add $t2, $t2, $s5 # 9
sub $t9, $t9, $t6 # 10

bgez $t9, krok14 # 11

add $t3, $t3, $s6 # 12
add $t9, $t9, $t6 # 13

krok14:

jal zamaluj # 14

sub $t0, $t0, 1 # dekrementacja licznika

b krok8

krok16:
cout ("\nkrok 16\n")
```

```
move $t9, $t6
div $t9, $t9, 2

move $t0, $t6

krok17:
bgtz $t0, krok18_23
b przewin

krok18_23:
add $t3, $t3, $s6 # 18

sub $t9, $t9, $t5 # 19

bgez $t9, krok23  # 20

add $t2, $t2, $s5 # 21

add $t9, $t9, $t6 # 22

krok23:
jal zamaluj # 23

sub $t0, $t0, 1 # dekrementacja licznika

b krok17


przewin: # przewiniecie $s3 tak by wskazywalo na poczatek kolejnej operacji
lb $t9, ($s3) # $s3 wskazuje na aktualny znak
beq $t9, $zero koniec
beq $t9, 13, czykoniec # patrzy czy koniec linii
add $s3, $s3, 1
b przewin

czykoniec: # czy koniec linii
add $s3, $s3, 1
lb $t9, ($s3)
beq $t9, 10, nowes3 # ustawia $s3 na poczatek kolejnej linii
b przewin

nowes3: # ustawienie $s3
add $s3, $s3, 1
b czyustaw

obrot: # obracanie zolwia o podany kat

jal czytajliczbe # czytanie o jaki kat nalezy obrocic
add $t4, $t4, $t0 # $t4 to aktualny kat ( dodajemy podany kat do niego )
bgt $t4, 359, obetnijkat # obcinanie kata w przypadku za duzego kata
```

```
cout ("\nkat wynosi tyle:\n")
move $a0, $t4
li $v0, 1
syscall

b przewin


obetnijkat: # obcinanie kata jesli wiekszy rowny 360 stopni

addiu $t4, $t4, -360

jr $ra


czytajliczbe: # czytanie liczby
# do zapisu liczby sluzy tutaj $t0
# $t9 wskazuje na aktualny znak

li $t0, 0
li $t9, 0

dodajczescliczby: # przechodzenie po znaku i dodawanie czesci liczby

lb $t9, ($s5)
blez $t9, powrot
beq $t9, ' ', powrot # sprawdzenie czy to liczba czy nie
blt $t9, '0' powrot
bgt $t9, '9', powrot


addiu $t9, $t9, -48 # dodanie czesci liczby ( nie ascii tylko od 0 do 9 ) dlatego odjete 4
mul $t0, $t0, 10 # mnozenie aktualnej liczby
add $t0, $t0, $t9 # dodanie nowej czesci zawartej w $t9
add $s5, $s5, 1 # przejscie do kolejnego znaku


b dodajczescliczby

powrot:
jr $ra


tworziwykonaj: # tworzenie polecen i ich wykonywanie ( przekierowanie do innych czesci pro
cout("tworze i wykonuje polecenia\n")

la $s3, tabela # ustawienie poczatkowych zawartosci $s3 i $s6 tak by wkazywaly odpowiednio
la $s6, tabela2
```

```
# 0 - opuszczony
# 1 - podniesiony
li $s0, 0 # domyslnie opuszczony !!!

b czyustaw
#b koniec
czyustaw: # sprawdzenie czy ma nastapic ustawienie zolwia
#cout ("\nczyustaw\n")

move $s5, $s3
beq $s5, $zero koniec

lb $t9, ($s5) # $t9 sluzy do skakania po znakach i patrzenia czy zgadzaja sie by wykonano
bne $t9, 'u', czypodnies

lb $t9, 1($s5)
bne $t9, 's', czypodnies

lb $t9, 2($s5)
bne $t9, 't', czypodnies

lb $t9, 3($s5)
bne $t9, 'a', czypodnies

lb $t9, 4($s5)
bne $t9, 'w', czypodnies

lb $t9, 5($s5)
bne $t9, ' ', czypodnies


add $s5, $s5, 6
#cout ("\nustaw\n")
b ustaw

ustaw: # ustawienie zolwia
# wczytanie 3 liczb oznaczajacych polozenie zolwia ( $t2, $t3 ) i kat z jakim ma byc pochy

cout ("\nustawiam\n")
# wczytaj 1 liczbe
jal czytajliczbe
add $s5, $s5, 1
#cout ("wczytano liczbe 1\n")
move $t2, $t0

# wczytaj 2 liczbe
jal czytajliczbe
add $s5, $s5, 1
#cout ("wczytano liczbe 2\n")
```

```
move $t3, $t0

# wczytaj 3 liczbe
jal czytajliczbe
#cout ("wczytano liczbe 3\n")
move $t4, $t0

cout ("\nkat wynosi tyle:\n")
move $a0, $t4
li $v0, 1
syscall


#b koniec
b przewin
#b czyustaw


czypodnies: # sprawdzenie czy ma nastapic podniesienie zolwia
# %t9 sluzy do skakania po znakach

move $s5, $s3

lb $t9, ($s5)
bne $t9, 'p', czyopusc

lb $t9, 1($s5)
bne $t9, 'o', czyopusc

lb $t9, 2($s5)
bne $t9, 'd', czyopusc

lb $t9, 3($s5)
bne $t9, 'n', czyopusc

lb $t9, 4($s5)
bne $t9, 'i', czyopusc

lb $t9, 5($s5)
bne $t9, 'e', czyopusc

lb $t9, 6($s5)
bne $t9, 's', czyopusc


li $s0, 1 # ustawienie uniesienia na 1 ( $ s0 )
# 1 oznacza stan podniesiony
# 0 oznacza stan opuszczony

b przewin
```

```
czyopusc: # sprawdzenie czy ma nastapic opuszczenie zolwia
# %t9 sluzy do skakania po znakach
move $s5, $s3

lb $t9, ($s5)
bne $t9, 'o', czynaprzod

lb $t9, 1($s5)
bne $t9, 'p', czynaprzod

lb $t9, 2($s5)
bne $t9, 'u', czynaprzod

lb $t9, 3($s5)
bne $t9, 's', czynaprzod

lb $t9, 4($s5)
bne $t9, 'c', czynaprzod

li $s0, 0 # ustawienie uniesienia na 0 ( $ s0 )
# 1 oznacza stan podniesiony
# 0 oznacza stan opuszczony

b przewin
#b czyustaw

czynaprzod: # sprawdzenie czy ma nastapic poruszenie sie zolwia do przodu
# %t9 sluzy do skakania po znakach

#b koniec

move $s5, $s3

lb $t9, ($s5)
bne $t9, 'n', czyobrot

lb $t9, 1($s5)
bne $t9, 'a', czyobrot

lb $t9, 2($s5)
bne $t9, 'p', czyobrot

lb $t9, 3($s5)
bne $t9, 'r', czyobrot

lb $t9, 4($s5)
bne $t9, 'z', czyobrot

lb $t9, 5($s5)
```

```
bne $t9, 'o', czyobrot

lb $t9, 6($s5)
bne $t9, 'd', czyobrot

lb $t9, 7($s5)
bne $t9, ' ', czyobrot

add $s5, $s5, 8
#cout ("\nnaprzod\n")

#b koniec

jal czytajliczbe # czytanie liczby oznaczajacej o ile do przodu ma isc zolw
move $t6, $t0 # w t6 jest o ile do przodu

cout ("\no tyle do przodu\n")
move $a0, $t6
li $v0, 1
syscall

b naprzod
czyobrot: # sprawdzenie czy ma nastapic obrocenie zolwia
# %t9 sluzy do skakania po znakach

move $s5, $s3

lb $t9, ($s5)
bne $t9, 'o', zlainstrukcja

lb $t9, 1($s5)
bne $t9, 'b', zlainstrukcja

lb $t9, 2($s5)
bne $t9, 'r', zlainstrukcja

lb $t9, 3($s5)
bne $t9, 'o', zlainstrukcja

lb $t9, 4($s5)
bne $t9, 't', zlainstrukcja

lb $t9, 5($s5)
bne $t9, ' ', zlainstrukcja


add $s5, $s5, 6

b obrot # obracanie zolwia
```

```
zlainstrukcja: # wyswietlenie komunikatu informujacego o niewlasciwej instrukcji
cout ("\nNie ma takiej istrukcji !!!\n")
b przewin

koniec: # koniec dzialania programu
li $v0, 15
move $a0, $s7
la $a1, tab1
li $a2, 576000
syscall#dopisuje tablice pikseli do pliku

move $a0, $s7
li $v0, 16
syscall#zamknalem plik
cout("koniec\n")
li $v0, 10
syscall
```