

# Interfejsów Komputerów Cyfrowych

## 4. Komunikacja z adapterem urządzenia zobrazowania

Arkadiusz Ostrzyżek

WCY22KY2S1 83744

29/05/2024

Zadanie wykonane na ocene: 5

## Analiza budowy i działania programu a1.asm.

.286

.model small

.data

.stack 64

.code

start:

MOV AX, 0B800H;	-	ustawia AX na adres segmentu pamięci tekstowej dla trybu tekstowego wideo
MOV ES, AX	-	przenosi AX do rejestru dodatkowego
MOV ES:[240], 7C14H	-	ustawia wybraną wartość pod adresem 240.
MOV AH, 1H	-	ładuje wartość przerwania do AH
INT 21H	-	wykonuje przerwanie, odczytuje znak z klawiatury
MOV AX, 4C00H	-	ładuje wartość przerwania do AX
INT 21H	-	przerwanie kończy działanie programu
end start	-	etykieta końca programu

## Zadanie na ocene 3.

a) program l2dsta.asm

### Sformułowanie problemu:

program l2dsta.asm, powodujący wyświetlenie w miejsce dotychczasowego symbolu serca

=== "pi" ¶ studenci o numerach nieparzystych,

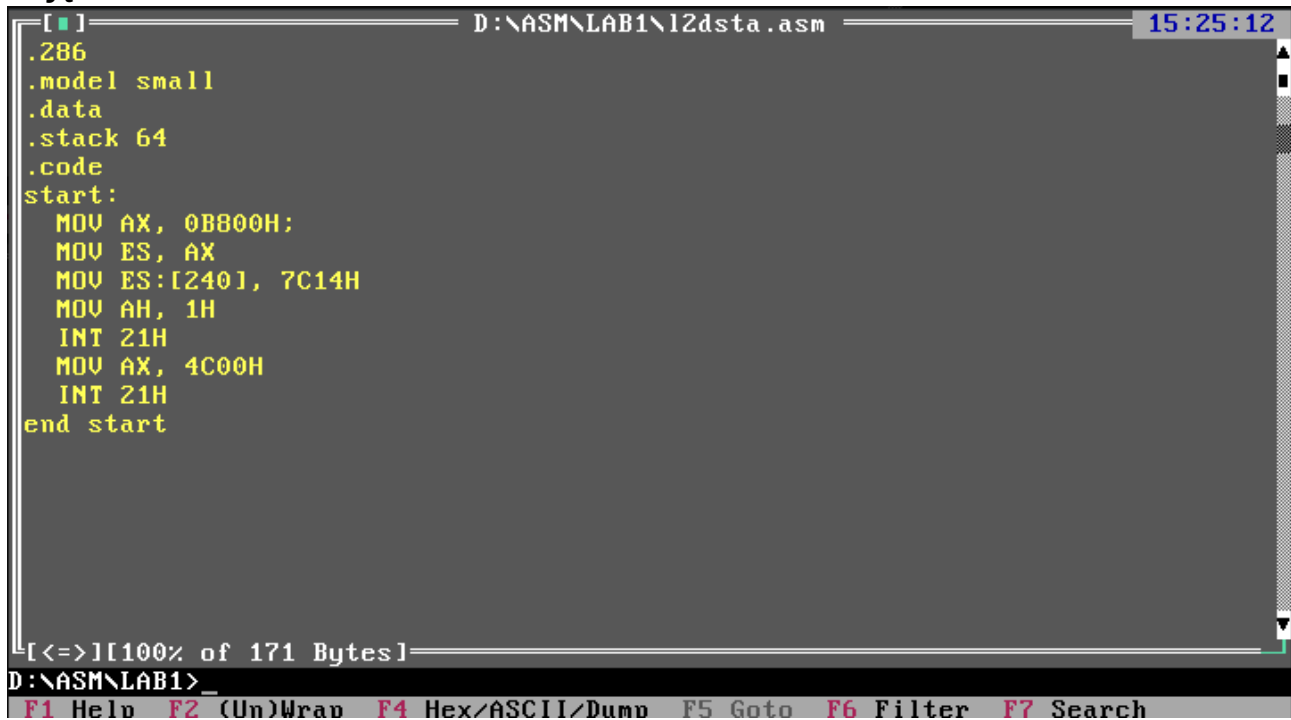
=== "paragraf" § studenci o numerach parzystych.

Pomysł na rozwiązanie problemu:

Musmy zmienić wartość podaną w hexie, która jest przesuwana w miejsce, gdzie uprzednio było serce. Kolor pozostaje ten sam, a więc zmieniamy tylko ostatni bajt na 14H.

MOV AX, 0B800H ładuje wartość 0B800H (adres segmentu pamięci tekstowej dla trybu tekstowego wideo) do rejestru AX.

## Zdjęcia kodu:



```
[ ] D:\ASM\LAB1\12dsta.asm 15:25:12
.286
.model small
.data
.stack 64
.code
start:
    MOV AX, 0B800H;
    MOV ES, AX
    MOV ES:[240], 7C14H
    MOV AH, 1H
    INT 21H
    MOV AX, 4C00H
    INT 21H
end start

[<=>][100% of 171 Bytes]
D:\ASM\LAB1>
F1 Help F2 (Un)Wrap F4 Hex/ASCII/Dump F5 Goto F6 Filter F7 Search
```

## Opis i efekty wykonania z rzutami ekranu:



## Zobrazowanie algorytmu:

b) program l2dstb.asm

## Sformułowanie problemu:

program **l2dstb.asm**, powodujący wyświetlenie napisu, składającego się z 5 powtórzeń wielkiej litery rozpoczynającej nazwisko wykonawcy. Napis ma rozpoczynać się w miejscu ekranu określonym współrzędnymi:

\* wiersz =  $6 + \text{nr w dzienniku}$ , kolumna =  $10 + \text{nr w dzienniku}$  studenci o numerach nieparzystych,

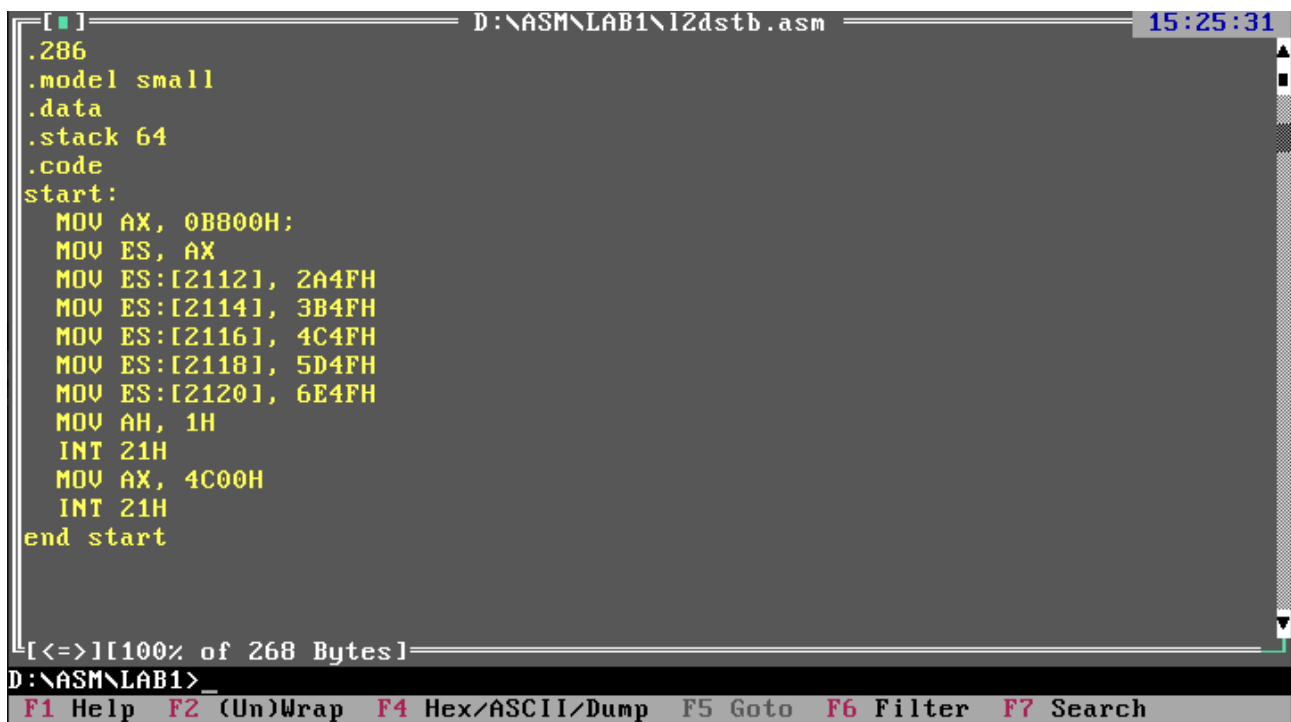
\* wiersz =  $8 + \text{nr w dzienniku}$ , kolumna =  $\text{nr w dzienniku}$  studenci o numerach parzystych. wykonawcy sprawozdania.

## Pomysł na rozwiązanie problemu:

Musimy zmienić miejsce, w którym znaki się wyświetlają, a więc zmieniamy pierwszy argument. Nowe dane obliczamy poprzez wykonanie działania  $(\text{wiersz} * 80 + \text{kolumna}) * 2 - 2$ . Tym razem musimy mieć unikatowe kolory tła i tekstu, a więc zmieniamy początkowy bajt, tak aby wszystkie półbajty były unikatowe, a drugi bajt ustawiamy na wartość odpowiadającą wybranemu przez nas znakowi. Wykonujemy tą samą linijkę z przeunięciem o 2 bajty parokrotnie.

MOV AX, 0B800H ładuje wartość 0B800H (adres segmentu pamięci tekstowej dla trybu tekstowego wideo) do rejestru AX.

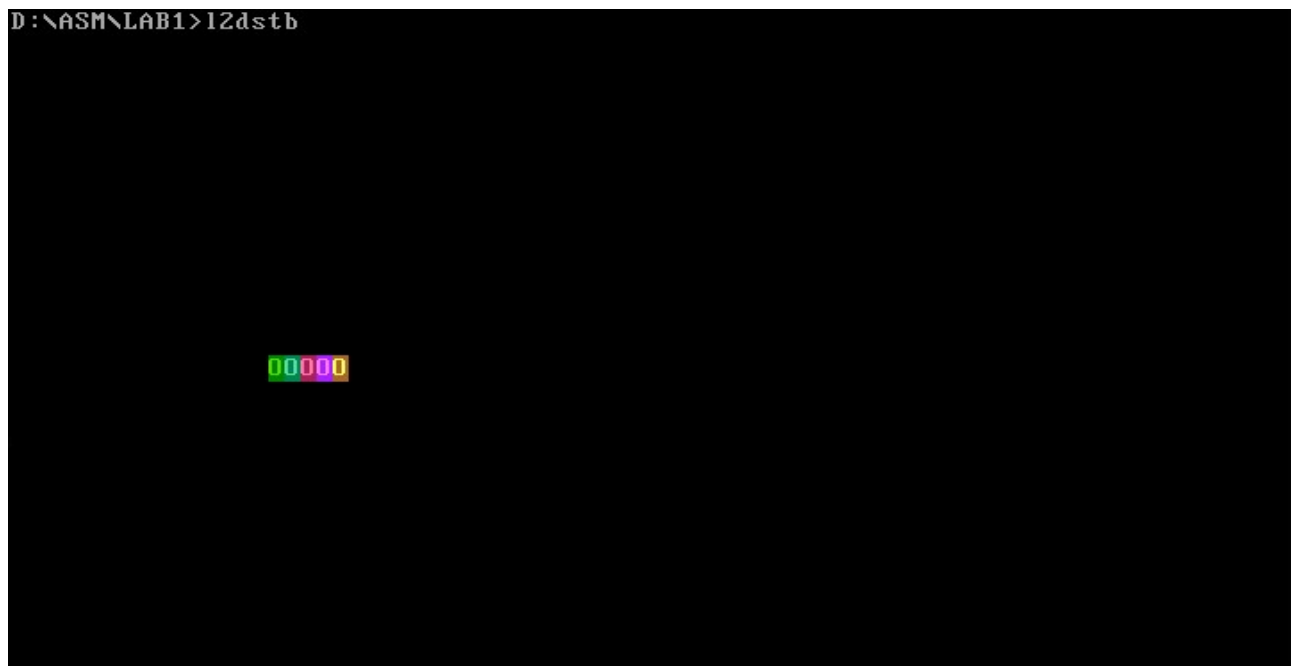
## Zdjęcia kodu:



```
[ ] D:\ASM\LAB1\l2dstb.asm 15:25:31
.286
.model small
.data
.stack 64
.code
start:
    MOV AX, 0B800H;
    MOV ES, AX
    MOV ES:[2112], 2A4FH
    MOV ES:[2114], 3B4FH
    MOV ES:[2116], 4C4FH
    MOV ES:[2118], 5D4FH
    MOV ES:[2120], 6E4FH
    MOV AH, 1H
    INT 21H
    MOV AX, 4C00H
    INT 21H
end start

[<=>] [100% of 268 Bytes]
D:\ASM\LAB1>_
F1 Help F2 (Un)Wrap F4 Hex/ASCII/Dump F5 Goto F6 Filter F7 Search
```

## Opis i efekty wykonania z rzutami ekranu:



## Zadanie na ocene 4.

### Sformułowanie problemu:

zmodyfikować uzyskany program, aby przy wykorzystaniu rozwiązania podobnego do tego z procedury "Rys" po uruchomieniu otrzymać 8 poziomych pasów tej samej wysokości każdy, zajmujących w sumie cały ekran, w kolorach od góry odpowiednio:

a) studenci o numerach nieparzystych odpowiednio: black, blue, green, cyan, dark gray, bright blue, bright green, bright cyan;

### Pomysł na rozwiązanie problemu:

W przykładowym kodzie można zauważyć, że DX wyznacza początek rysowania, a 50 wyznacza koniec rysowania. Musimy zmienić więc zmienić 50 na rejestr, np BX i zwiększać go przy rysowaniu każdego kolejnego paska.

### Pisemna analiza "ognia".

MOV AX, 0F00H, INT 10H – ustawia AX na 0F00H, czyli obecny tryb wideo.

MOV AX, 0013H, INT 10H – ustawia AX na 0013H i wywołuje przerwanie, które ustawia tryb wideo na graficzny 320x200 pixeli, 256 kolorów.

Następnie 8 razy powtarzany jest proces zwiększania BX, odpowiedzialnego za wiersz od którego zaczniemy rysować, o 25 (200/8), ustawiania rejestru na AL na wartość równą kolorowi w którym chcemy rysować i wywoływania procesu Rys.

MOV AX, 0F00H, INT 10H – ustawia AX na 0F00H, czyli obecny tryb wideo.


MOV AX, 0003H, INT 10H – ustawia AX na 0003H, czyli tryb tekstowy.

MOV AX, 4C00H, INT 21H – zakończenie programu

### Pisemna analiza "Rys".

JL oznacza Jump if less, a więc program wykona pierwszą część programu od etRys1 do jl 320 razy, ponieważ potem CX, czyli numer kolumny, byłby większy niż 320. Następnie DX, czyli numer wiersza po którym się poruszamy zostanie zwiększony o jeden i powrócimy do etRys1. Ta pętla powtórzy się tak długo, aż DX będzie większy od BX.

## Zdjęcia kodu:



The screenshot shows a window titled "D:\ASM\LAB1\rzs.asm" with a timestamp of "15:26:00". The code is written in x86 assembly language. It starts with ".286", ".model small", ".data", and ".code". The main code block is labeled "start:" and contains instructions for jumping to "ognia", defining a procedure "Rys", and a loop labeled "etRys1" that increments CX, compares it with 320, and jumps back to "etRys1" if less than. It also increments DX, compares it with 50 and BX, and jumps back to "etRys1" if less than. The code ends with "RET" and "ENDP". A label "ognia:" is also present. The status bar at the bottom shows "D:\ASM\LAB1>" and a list of function keys: F1 Help, F2 (Un)Wrap, F4 Hex/ASCII/Dump, F5 Goto, F6 Filter, F7 Search.

```
[ ] D:\ASM\LAB1\rzs.asm 15:26:00
.286
.model small
.data
.code
start:
    JMP ognia
    Rys PROC
    etRys1:
    INT 10H
    INC CX
    CMP CX, 320
    JL etRys1
    MOV CX, 0
    INC DX
;    CMP DX, 50
    CMP DX, BX
    JL etRys1
    RET
    ENDP

ognia:
[<=>]10% of 872 Bytes]
D:\ASM\LAB1>
F1 Help F2 (Un)Wrap F4 Hex/ASCII/Dump F5 Goto F6 Filter F7 Search
```

```
[■] D:\ASM\LAB1\rys.asm 15:26:23
ognia:
MOV AX, 0F00H
INT 10H
MOV AX, 0013H
INT 10H
MOV AH, 0CH
MOV CX, 0
MOV DX, 0

; black
MOV BX, 25
MOV AL, 0
CALL Rys

; blue
ADD BX, 25
MOV AL, 1
CALL Rys

; green
ADD BX, 25
[<=>] 123% of 872 Bytes]
D:\ASM\LAB1>
F1 Help F2 (Un)Wrap F4 Hex/ASCII/Dump F5 Goto F6 Filter F7 Search
```

```
[■] D:\ASM\LAB1\rys.asm 15:26:43
; green
ADD BX, 25
MOV AL, 2
CALL Rys

; cyan
ADD BX, 25
MOV AL, 3
CALL Rys

; dark gray
ADD BX, 25
MOV AL, 8
CALL Rys

; bright blue
ADD BX, 25
MOV AL, 9
CALL Rys

; bright green
[<=>] 147% of 872 Bytes]
D:\ASM\LAB1>
F1 Help F2 (Un)Wrap F4 Hex/ASCII/Dump F5 Goto F6 Filter F7 Search
```



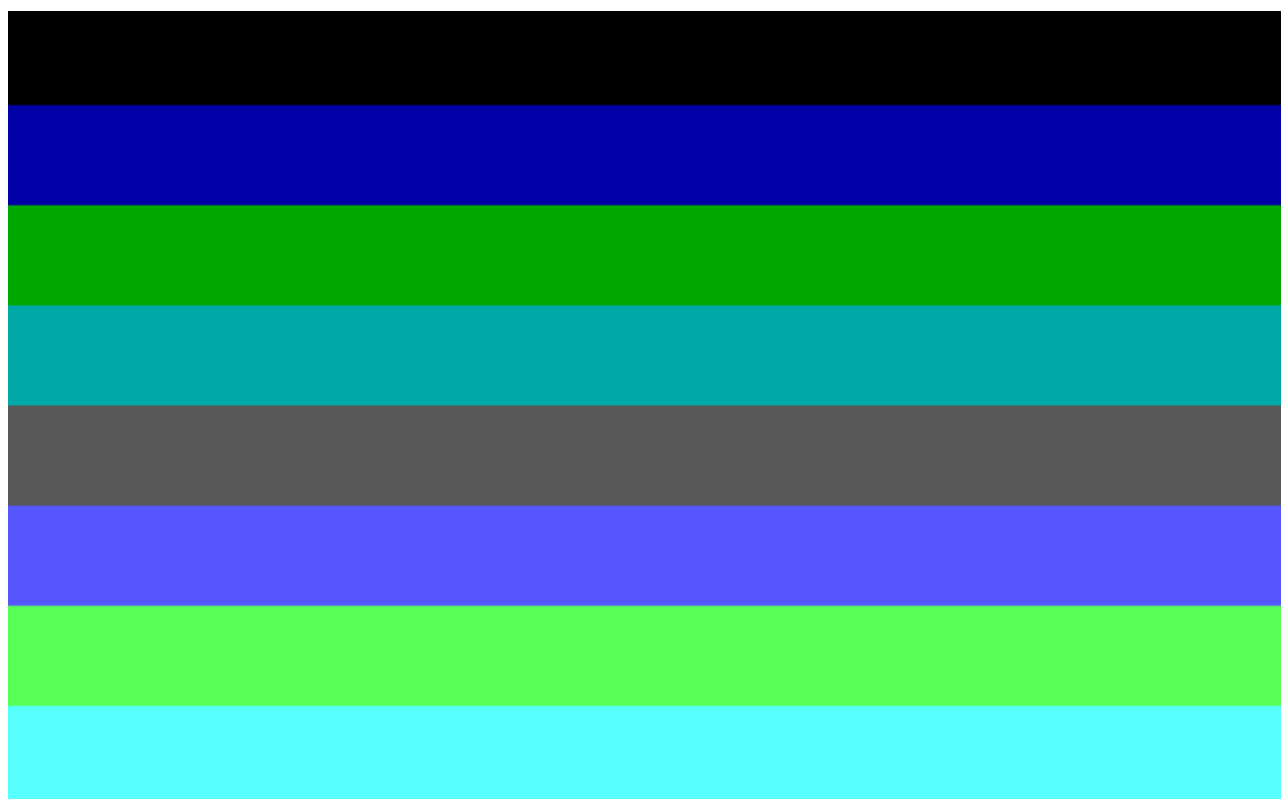
```
[■] D:\ASM\LAB1\rys.asm 15:27:00
; bright green
ADD BX, 25
MOV AL, 10
CALL Rys

; bright cyan
ADD BX, 25
MOV AL, 11
CALL Rys

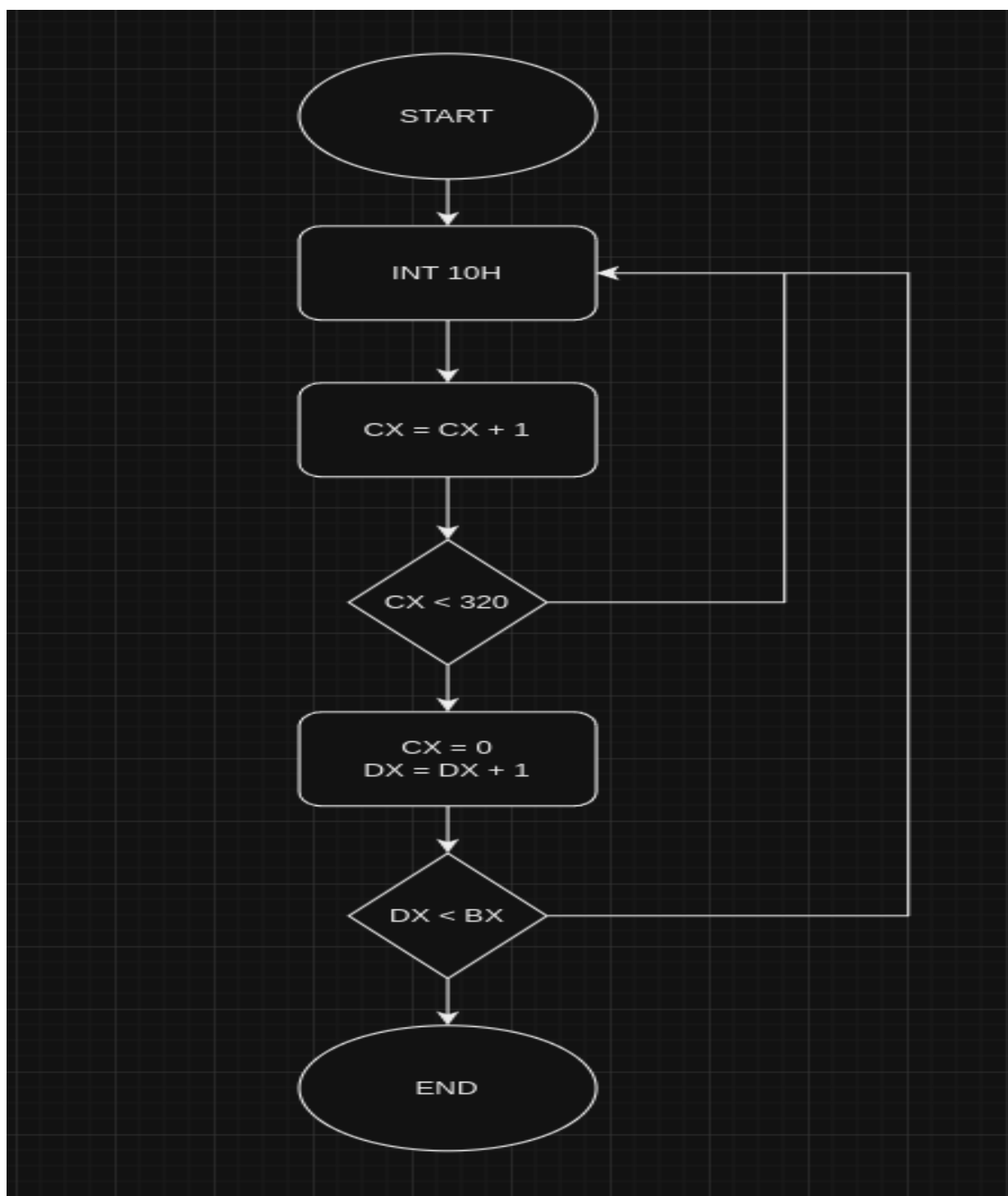
MOV AH, 08H
INT 21H
MOV AX, 0F00H
INT 10H
MOV AX, 0003H
INT 10H
MOV AX, 4C00H
INT 21H
.stack 100H
end start

[<=>][100% of 872 Bytes]
D:\ASM\LAB1>_
F1 Help F2 (Un)Wrap F4 Hex/ASCII/Dump F5 Goto F6 Filter F7 Search
```

Opis i efekty wykonania z rzutami ekranu:



Zobrazowanie algorytmu:



## Zadanie na ocene 5.

### Sformułowanie problemu:

Zmodyfikować przedstawiony program w opisanej części, "deaktywując przez znak komentarza na początku linii" deklarację i wywołanie procedury "Rys", podobnie do działań na ocenę dobrze, a następnie tak zmodyfikować uzyskany program, aby przy wykorzystaniu rozwiązania podobnego do tego z procedury "Rys2" po uruchomieniu otrzymać 8 poziomych pasów tej samej wysokości każdy, zajmujących w sumie cały ekran, w kolorach **od góry odpowiednio**:

a) studenci o numerach nieparzystych odpowiednio: red, magenta, brown, white(light gray), bright red, bright magenta, bright yellow, bright white;

### Pomysł na rozwiązanie problemu:

W przykładowym kodzie można zauważyć, że MOV BX 0 wyznacza początek rysowania, a CMP BX, 32000 wyznacza koniec rysowania. Musimy zmienić więc te wartości na jakieś rejestry, abyśmy mogli modyfikować przed wywołaniem programu, gdzie rozpoczniemy i zakończymy rysowanie. Kolor, którym chcemy rysować też musi być gdzieś przechowywany i został do tego wybrany rejestr BL, który jest ustawiany na CX, po przekazaniu informacji.

### Pisemna analiza "ognia":

MOV AX, 0F00H, INT 10H – ustawia AX na 0F00H, czyli obecny tryb wideo.

MOV AX, 0013H, INT 10H – ustawia AX na 0013H i wywołuje przerwanie, które ustawia tryb wideo na graficzny 320x200 pixeli, 256 kolorów.

Następnie 8 razy powtarzany jest proces zwiększania BX, odpowiedzialnego za wiersz od którego zaczniemy rysować, o 25 (200/8), ustawiania rejestru na AL na wartość równą kolorowi w którym chcemy rysować i wywoływania procesu Rys.

MOV AX, 0F00H, INT 10H – ustawia AX na 0F00H, czyli obecny tryb wideo.

MOV AX, 0003H, INT 10H – ustawia AX na 0003H, czyli tryb tekstowy.

MOV AX, 4C00H, INT 21H – zakończenie programu

### Pisemna analiza "Rys2".

MOV AX, 0A000H – ustawia AX na początkowy adres pamięci graficznej VGA.

Ustawiamy potem ES na AX. Następnie rozpoczyna się główna pętla Rys2.

Ustawiamy wartość piksela na pozycji startowej + bx na wartość równą wybranemu przez nas kolorowi. Zwiększamy BX, aby wskazywać na kolejny

pixel. Jeżeli nie dotarliśmy jeszcze do pixela oznaczonego w BX, to powtarzamy czynność.

## Zdjęcia kodu:

```
[ ] D:\ASM\LAB1\rys2.asm 15:27:34
.286
.model small
.data
.code
start:
    JMP ognia

    Rys2 PROC
    MOV AX, 0A000H
    MOV ES, AX
    MOV AL, BL
    MOV BX, CX
    etRys2:
    MOV ES:[BX], AL
    INC BX
    CMP BX, DX
    JB etRys2
    RET
    ENDP

    ognia:
[<=>][0% of 1,041 Bytes]
D:\ASM\LAB1>
F1 Help F2 (Un)Wrap F4 Hex/ASCII/Dump F5 Goto F6 Filter F7 Search
```

```
[ ] D:\ASM\LAB1\rys2.asm 15:27:54
    ognia:
    MOV AX, 0F00H
    INT 10H
    MOV AX, 0013H
    INT 10H
    MOV AH, 0CH
    MOV CX, 0
    MOV DX, 0
    MOV AL, 15

; red
    MOV BL, 4
    MOV CX, 0
    MOV DX, 8000
    CALL Rys2

; magneta
    MOV BL, 5
    MOV CX, DX
    ADD DX, 8000
    CALL Rys2
[<=>][20% of 1,041 Bytes]
D:\ASM\LAB1>
F1 Help F2 (Un)Wrap F4 Hex/ASCII/Dump F5 Goto F6 Filter F7 Search
```

```
[■] D:\ASM\LAB1\rys2.asm 15:28:25
; brown
MOV BL, 6
MOV CX, DX
ADD DX, 8000
CALL Rys2

; light gray
MOV BL, 7
MOV CX, DX
ADD DX, 8000
CALL Rys2

; bright red
MOV BL, 12
MOV CX, DX
ADD DX, 8000
CALL Rys2

; bright magneta
MOV BL, 13
MOV CX, DX
CALL Rys2
[<=>] 44% of 1,041 Bytes]
D:\ASM\LAB1>
F1 Help F2 (Un)Wrap F4 Hex/ASCII/Dump F5 Goto F6 Filter F7 Search
```

```
[■] D:\ASM\LAB1\rys2.asm 15:28:46
; bright magneta
MOV BL, 13
MOV CX, DX
ADD DX, 8000
CALL Rys2

; bright yellow
MOV BL, 14
MOV CX, DX
ADD DX, 8000
CALL Rys2

; bright white
MOV BL, 15
MOV CX, DX
ADD DX, 8000
CALL Rys2

MOV AH, 08H
INT 21H
MOV AX, 0F00H
[<=>] 65% of 1,041 Bytes]
D:\ASM\LAB1>
F1 Help F2 (Un)Wrap F4 Hex/ASCII/Dump F5 Goto F6 Filter F7 Search
```

```
[ ] D:\ASM\LAB1\rzs2.asm 15:28:58
MOV CX, DX
ADD DX, 8000
CALL Rzs2

; bright white
MOV BL, 15
MOV CX, DX
ADD DX, 8000
CALL Rzs2

MOV AH, 08H
INT 21H
MOV AX, 0F00H
INT 10H
MOV AX, 0003H
INT 10H
MOV AX, 4C00H
INT 21H
.stack 100H
end start

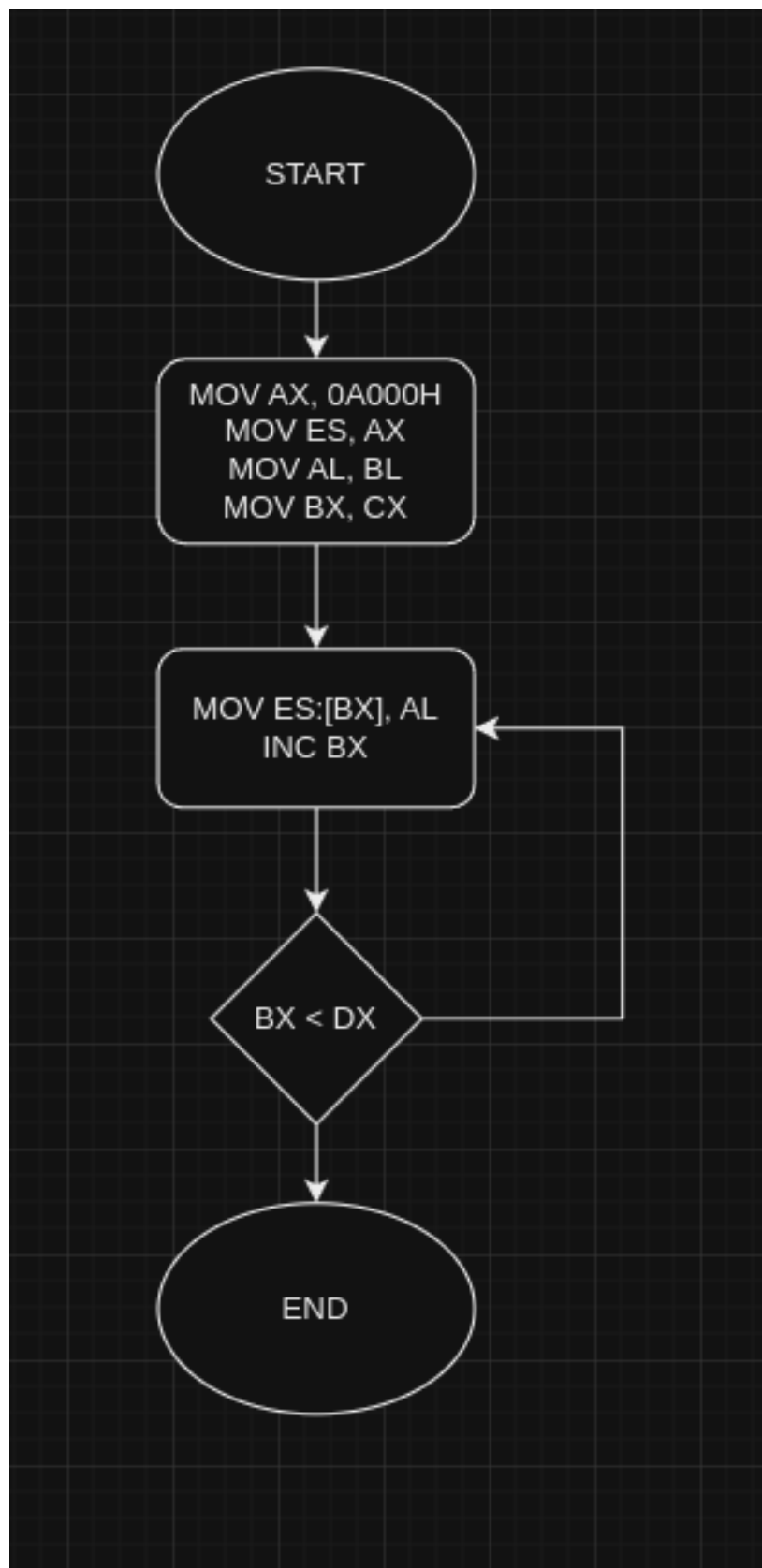
[<=>][100% of 1,041 Bytes]
D:\ASM\LAB1>
F1 Help F2 (Un)Wrap F4 Hex/ASCII/Dump F5 Goto F6 Filter F7 Search
```

Opis i efekty wykonania z rzutami ekranu:



(Na dole jest biały)

Zobrazowanie algorytmu:



Zdjęcie poprawnej kompilacji każdego z programów:

```
D:\ASM\LAB1>comp l2dsta 15:23:16
D:\ASM\LAB1>tasm l2dsta.asm
Turbo Assembler Version 4.1 Copyright (c) 1988, 1996 Borland International

Assembling file: l2dsta.asm
Error messages: None
Warning messages: None
Passes: 1
Remaining memory: 348k

D:\ASM\LAB1>tlink /v l2dsta.obj
Turbo Link Version 7.1.30.1. Copyright (c) 1987, 1996 Borland International

D:\ASM\LAB1>
D:\ASM\LAB1>_
F1 Help F10 Menu
```

```
D:\ASM\LAB1>comp l2dstb 15:23:58
D:\ASM\LAB1>tasm l2dstb.asm
Turbo Assembler Version 4.1 Copyright (c) 1988, 1996 Borland International

Assembling file: l2dstb.asm
Error messages: None
Warning messages: None
Passes: 1
Remaining memory: 348k

D:\ASM\LAB1>tlink /v l2dstb.obj
Turbo Link Version 7.1.30.1. Copyright (c) 1987, 1996 Borland International

D:\ASM\LAB1>
D:\ASM\LAB1>
F1 Help F10 Menu
```



```
D:\ASM\LAB1>comp rys 15:24:17
D:\ASM\LAB1>tasm rys.asm
Turbo Assembler Version 4.1 Copyright (c) 1988, 1996 Borland International

Assembling file:    rys.asm
Error messages:     None
Warning messages:   None
Passes:             1
Remaining memory:   347k

D:\ASM\LAB1>tlink /v rys.obj
Turbo Link Version 7.1.30.1. Copyright (c) 1987, 1996 Borland International

D:\ASM\LAB1>

D:\ASM\LAB1>
F1 Help  F10 Menu
```

```
D:\ASM\LAB1>comp rys 15:24:27
D:\ASM\LAB1>tasm rys.asm
Turbo Assembler Version 4.1 Copyright (c) 1988, 1996 Borland International

Assembling file:    rys.asm
Error messages:     None
Warning messages:   None
Passes:             1
Remaining memory:   347k

D:\ASM\LAB1>tlink /v rys.obj
Turbo Link Version 7.1.30.1. Copyright (c) 1987, 1996 Borland International

D:\ASM\LAB1>

D:\ASM\LAB1>
F1 Help  F10 Menu
```

## Programy w formie tekstowej:

### program l2dsta.asm

```
.286
.model small
.data
.stack 64
.code
start:
    MOV AX, 0B800H;
    MOV ES, AX
    MOV ES:[240], 7C14H
    MOV AH, 1H
    INT 21H
    MOV AX, 4C00H
    INT 21H
end start
```

### program l2dstab.asm

```
.286
.model small
.data
.stack 64
.code
start:
    MOV AX, 0B800H;
    MOV ES, AX
    MOV ES:[2112], 2AF4H
    MOV ES:[2114], 3BF4H
    MOV ES:[2116], 4CF4H
    MOV ES:[2118], 5DF4H
    MOV ES:[2120], 6EF4H
    MOV AH, 1H
    INT 21H
    MOV AX, 4C00H
    INT 21H
end start
```

## rys.asm

```
.286
.model small
.data
.code
```

start:

```
    JMP ognia
    Rys PROC
etRys1:
    INT 10H
    INT CX
    CMP CX, 320
    JL etRys1
    MOV CX, 0
    INC DX
;    CMP DX, 50
    CMP DX, BX
    JL etRys1
    RET
    ENDP
```

ognia:

```
    MOV AX, 0F00H
    INT 10H
    MOV AX, 0013H
    INT 10H
    MOV AH, 0CH
    MOV CX, 0
    MOV DX, 0
```

```
    MOV BX, 25
    MOV AL, 0
    CALL Rys
```

```
    ADD BX, 25
    MOV AL, 1
    CALL Rys
```

```
    ADD BX, 25
    MOV AL, 2
```

CALL Rys

ADD BX, 25

MOV AL, 3

CALL Rys

ADD BX, 25

MOV AL, 8

CALL Rys

ADD BX, 25

MOV AL, 9

CALL Rys

ADD BX, 25

MOV AL, 10

CALL Rys

ADD BX, 25

MOV AL, 11

CALL Rys

MOV AH, 08H

INT 21H

MOV AX, 0F00H

INT 10H

MOV AX, 0003H

INT 10H

MOV AX, 4C00H

INT 21H

.stack 100H

end start

## program rys2.asm

.286

.model small

.data

.code

start:

JMP ognia

Rys2 PROC

MOV AX, 0A000H

MOV ES, AX

MOV AL, BL

MOV BX, CX

etRys2:

MOV ES:[BX], AL

INC BX

CMP BX, DX

JB etRys2

RET

ENDP

ognia:

MOV AX, 0F00H

INT 10H

MOV AX, 0013H

INT 10H

MOV AH, 0CH

MOV CX, 0

MOV DX, 0

MOV BL, 4

MOV CX, 0

MOV DX, 8000

CALL Rys2

MOV BL, 5

MOV CX, DX

ADD DX, 8000

CALL Rys2

```
MOV BL, 6
MOV CX, DX
ADD DX, 8000
CALL Rys2
```

```
MOV BL, 7
MOV CX, DX
ADD DX, 8000
CALL Rys2
```

```
MOV BL, 12
MOV CX, DX
ADD DX, 8000
CALL Rys2
```

```
MOV BL, 13
MOV CX, DX
ADD DX, 8000
CALL Rys2
```

```
MOV BL, 14
MOV CX, DX
ADD DX, 8000
CALL Rys2
```

```
MOV BL, 15
MOV CX, DX
ADD DX, 8000
CALL Rys2
```

```
MOV AH, 08H
INT 21H
MOV AX, 0F00H
INT 10H
MOV AX, 0003H
INT 10H
MOV AX, 4C00H
INT 21H
.stack 100H
```

```
end start
```