

The background is a light blue sky with a large yellow sun in the top right corner. There are three white, fluffy clouds. The bottom of the image shows a green rolling landscape with two stylized green trees on the left and two on the right, with small pink flowers scattered on the grass.

KA. Pratybos 2/3

I.Grinis

CPU sandara (registrai, jų kodas)

16-Bit ($w = 1$)		8-Bit ($w = 0$)		Segment	
000	AX	000	AL	00	ES
001	CX	001	CL	01	CS
010	DX	010	DL	10	SS
011	BX	011	BL	11	DS
100	SP	100	AH		
101	BP	101	CH		
110	SI	110	DH		
111	DI	111	BH		

CPU sandara (bendrieji registrai)

AH	AL
BH	BL
CH	CL
DH	DL
SP	
BP	
SI	
DI	

AX

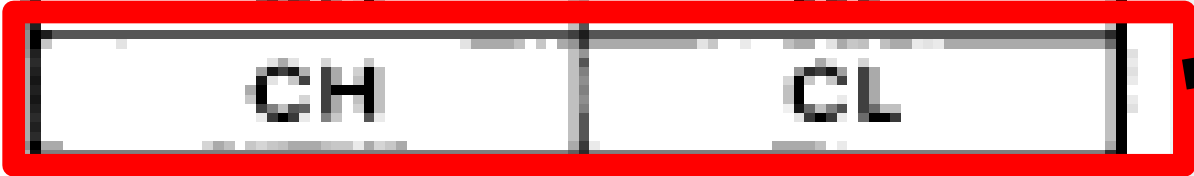
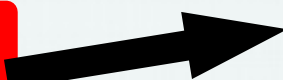
CPU sandara (bendrieji registrai)

AH	AL
BH	BL
CH	CL
DH	DL
SP	
BP	
SI	
DI	

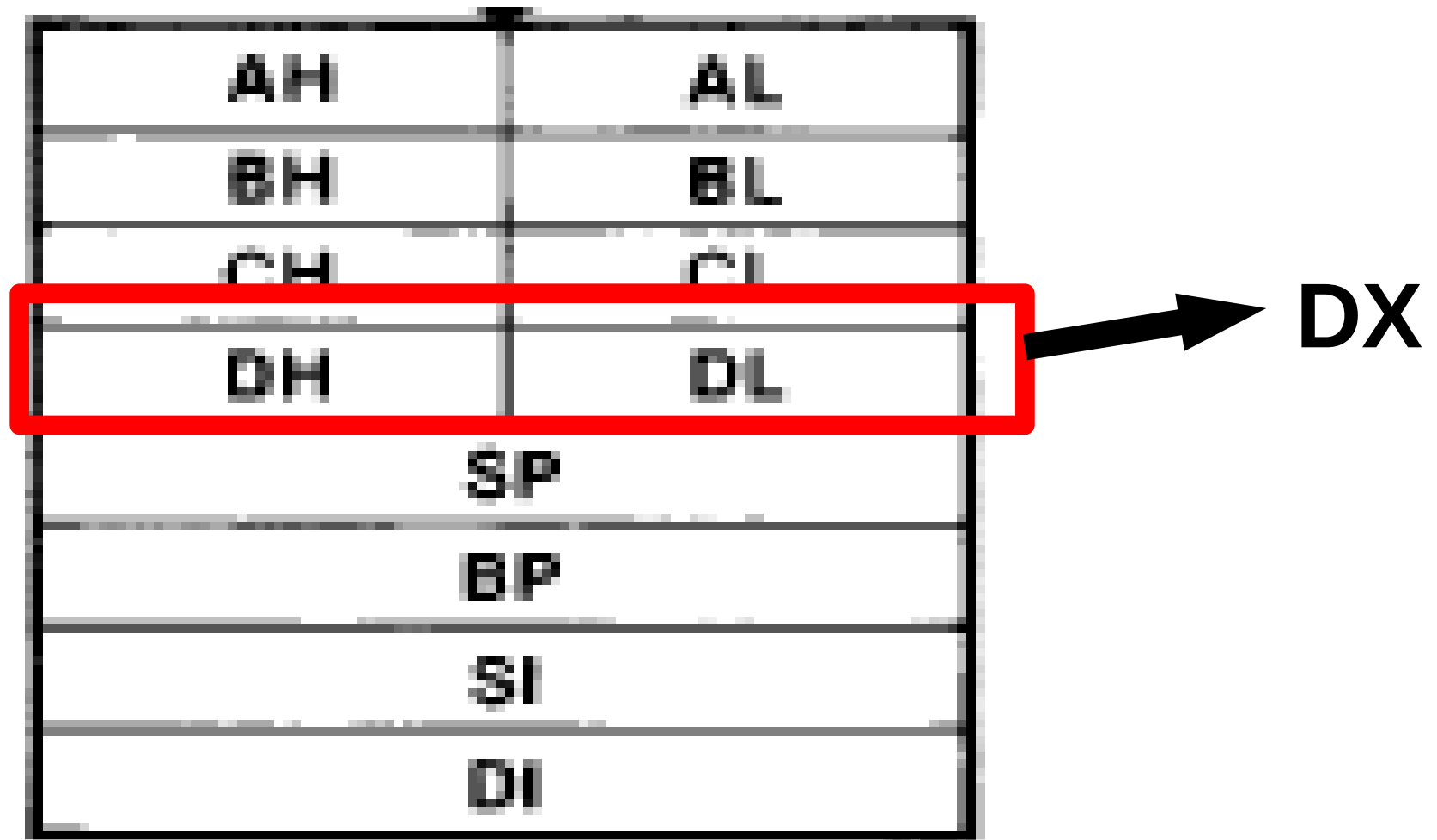
BX

CPU sandara (registrai, jų kodas)

AH	AL
BH	BL
CH	CL
DH	DL
SP	
BP	
SI	
DI	

  CX

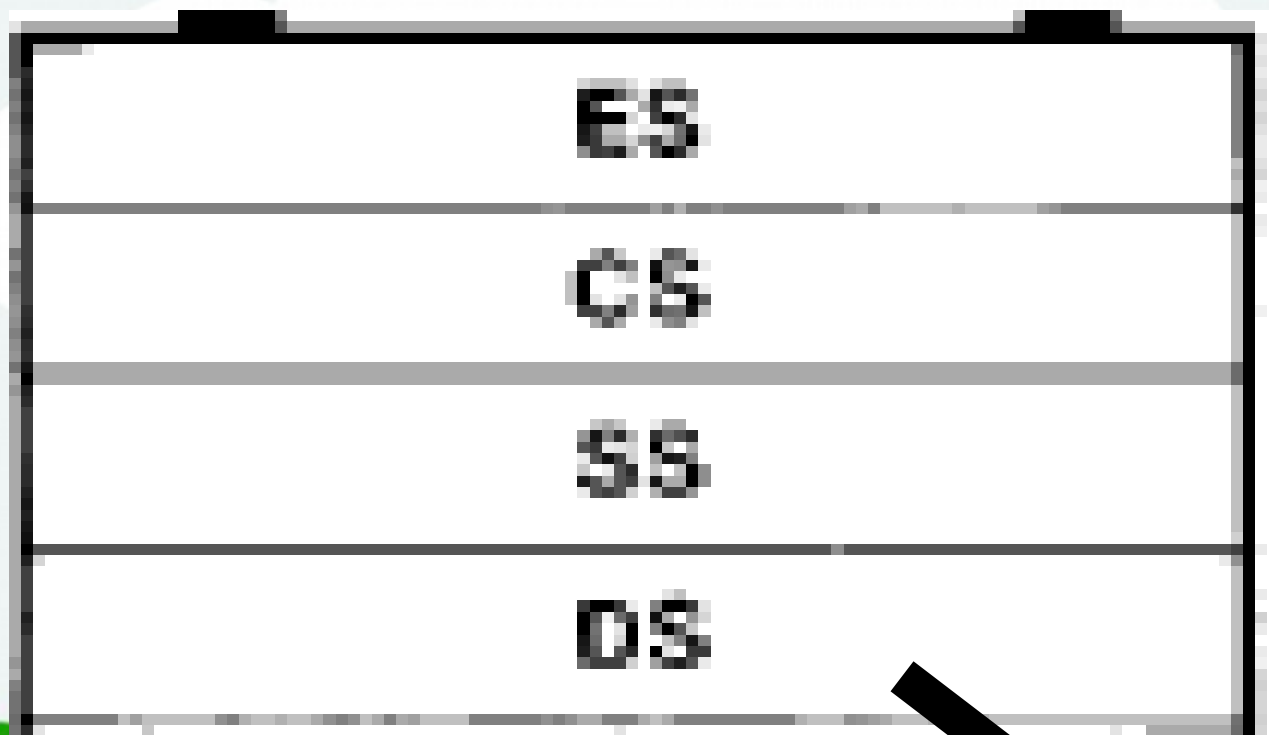
CPU sandara (registrai, jų kodas)



Segmentiniai registrai

ES
CS
SS
DS

Segmentiniai registrai



Duomenų
segmentas

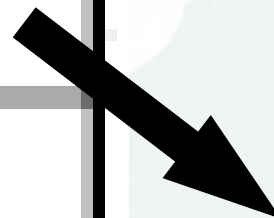
Segmentiniai registrai

ES
CS
SS
DS

Steko segmentas

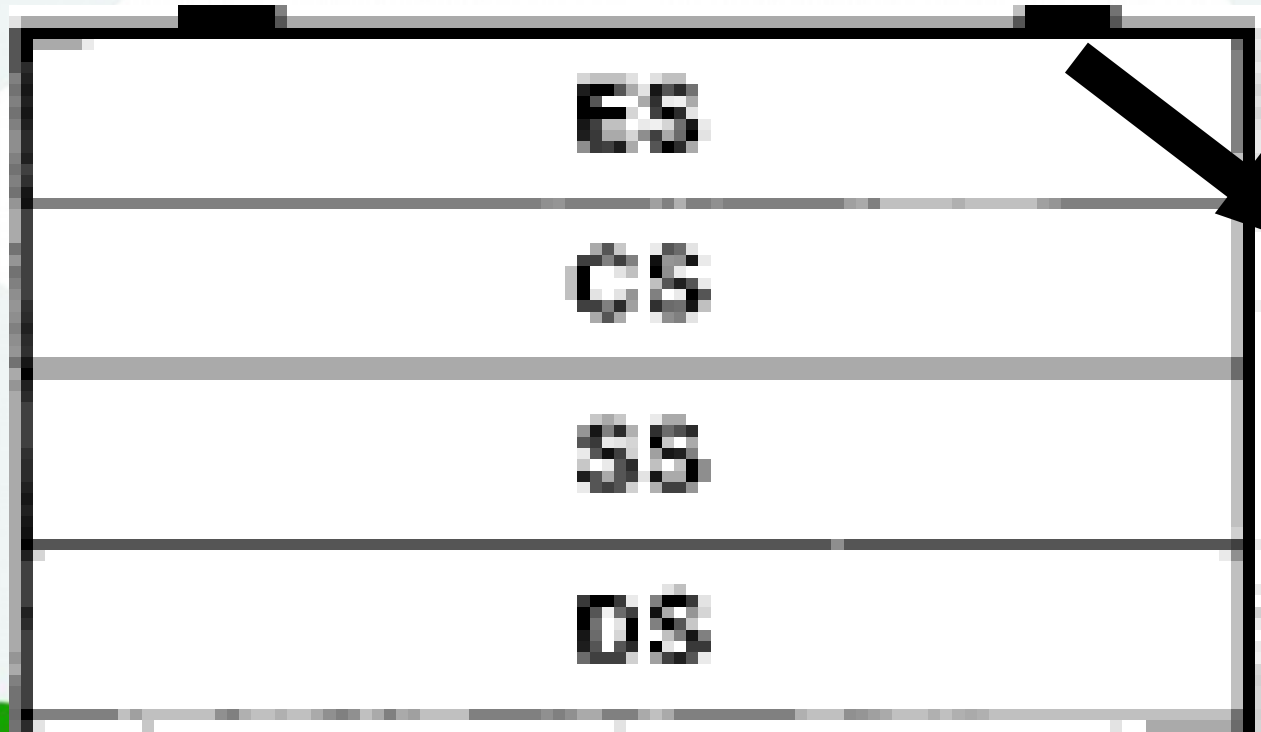
Segmentiniai registrai

ES
CS
SS
DS



Kodo segmentas

Segmentiniai registrai



**Papildomas
(duomenų)
segmentas**

Kaip formuojasi adresas?

- Klasikiniame PC gali būti iki 1 MB atminties, t. y., programuotojas turi gebėti adresuoti nuo 00000 iki FFFFFF.
- Adresas klasikiniame PC formuojasi iš dviejų dalių:
 - Segmento
 - Poslinkio
- Žymėjimas: Segmentas : Poslinkis
- Segmentas – tai atminties atkarpa iki 64 KB, kuri prasideda nuo adreso, kuris dalijasi iš 16.
- Poslinkis nurodo vietą SEGMENTE.
- Pilnas adresas skaičiuojasi pagal formulę:
 - Adresas = Segmentas * 16 + Poslinkis.

KA-Z

Kaip formuojasi adresas?

- Pavyzdys. Adresas 0000 : 1A11 yra fiziškai 01A11, o A000 : 1234 yra A1234.
- Pastaba vieną ir tą patį fizikinį adresą gali reikšti skirtingi žymėjimai:
 - $ABCDE = ABCD : 000E = ABC0 : 00DE = \dots$

Svarbu žinoti

- Duomenys paprastai imami iš segmento, kurio reikšmė įrašyta registre DS, **kitos** po vykdomos instrukcijos adresą rodo pora CS : IP, kur CS – kodo segmento registras, o **IP** – poslinkis kodo segmente.
- Vykdančios programos labai praverčia turėti specialią atmintį, kurioje galima laikinai saugoti įvairią informaciją. Tam reikalingas stekas, kuris naudojamas vykdančios tam tikras instrukcijas. Su steku susieti registrai yra SS (segmento) ir SP bei BP.

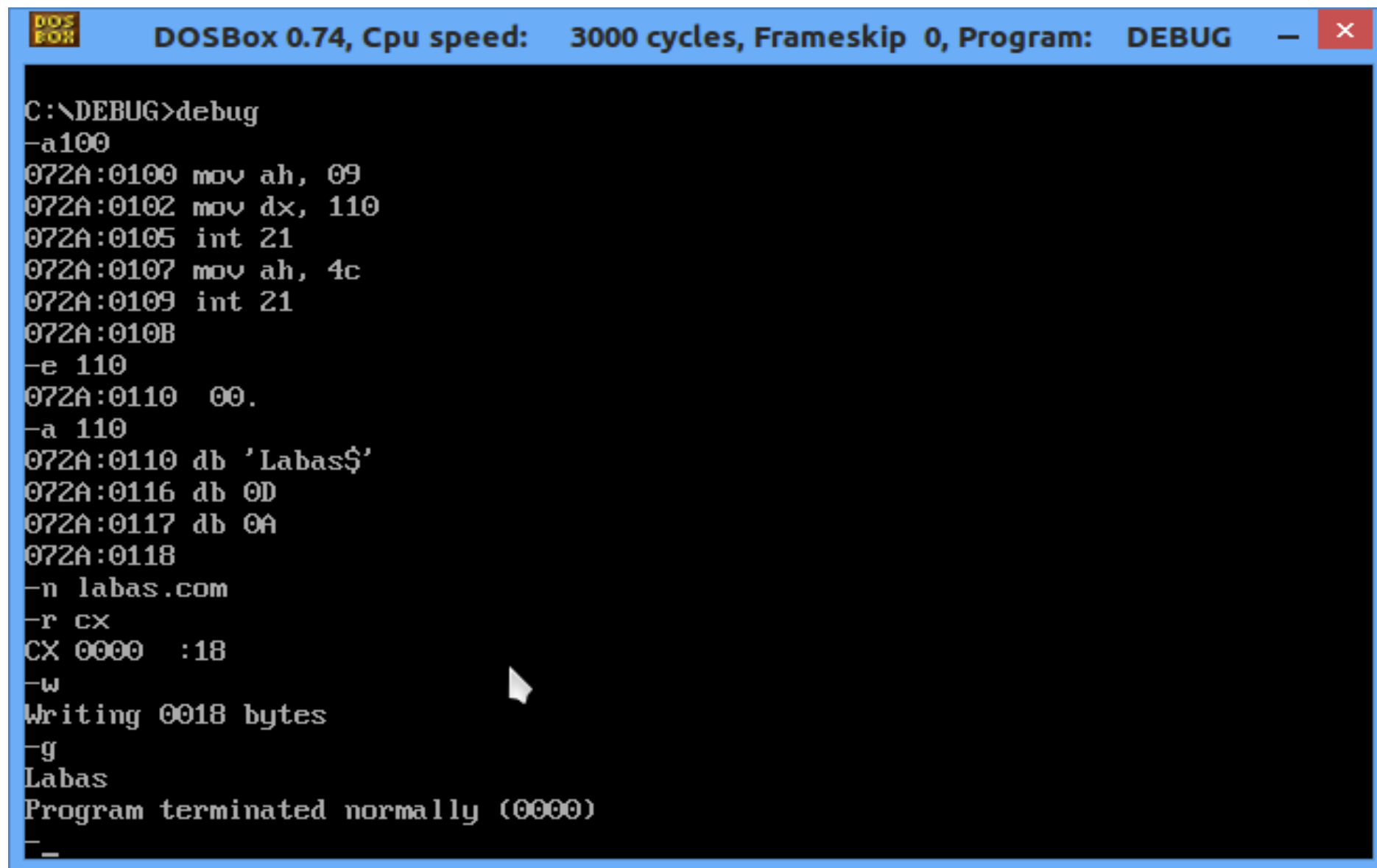
Debug programos nagrinėjimas

- DOS programa debug leidžia vykdyti pažingsniui ir redaguoti nesudėtingas mašininio kodo programas. Savo viduje ji turi disassemblerį, kuris leidžia programuotojui nesirūpinti dėl instrukcijų kodo nežinojimo.

Debug panaudojimas kuriant paprastą programą

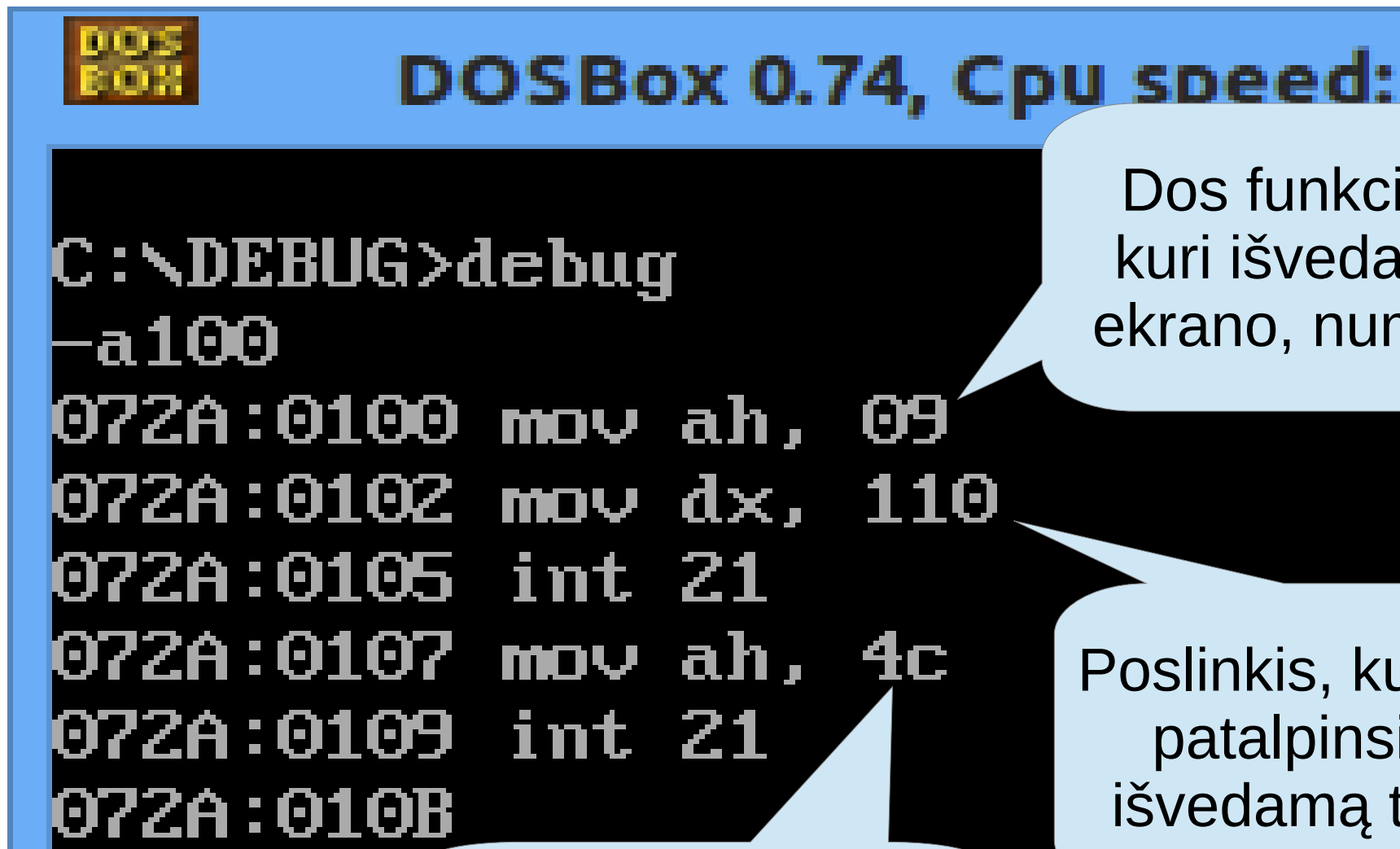
- Paleidus debug, galima panaudoti komandą *a<adresas>* instrukcijų įvedimui. *Adresas* gali būti pilnas arba tik su poslinkio dalimi. Pvz., *a100* → leis įvedinėti instrukcijas nuo adreso CS:0100, o *a1000:1234* → nuo adreso 1000:1234.

Programos įvedimo pavyzdys



```
DOS FOR
DOSBox 0.74, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
C:\DEBUG>debug
-a100
072A:0100 mov ah, 09
072A:0102 mov dx, 110
072A:0105 int 21
072A:0107 mov ah, 4c
072A:0109 int 21
072A:010B
-e 110
072A:0110 00.
-a 110
072A:0110 db 'Labas$'
072A:0116 db 0D
072A:0117 db 0A
072A:0118
-n labas.com
-r cx
CX 0000 :18
-w
Writing 0018 bytes
-g
Labas
Program terminated normally (0000)
_
```

1 dalis: įvedame kodą



The screenshot shows the DOSBox 0.74 interface with a blue title bar. On the left, there is a yellow 'DOS BOX' logo. The main window has a black background with white text. The text shows a command prompt where 'debug' has been entered, followed by a memory dump starting at address 072A:0100. The assembly instructions are: 'mov ah, 09', 'mov dx, 110', 'int 21', 'mov ah, 4c', 'int 21', and '072A:010B'. Three light blue speech bubbles with black text provide explanations for specific instructions: one for 'int 21' at 0100, one for 'int 21' at 0109, and one for 'mov ah, 4c'.

```
DOS BOX    DOSBox 0.74, Cpu speed:

C:\DEBUG>debug
-a100
072A:0100  mov ah, 09
072A:0102  mov dx, 110
072A:0105  int 21
072A:0107  mov ah, 4c
072A:0109  int 21
072A:010B
```

Dos funkcijos, kuri išveda ant ekrano, numeris

Poslinkis, kuriame patalpinsime išvedamą tekstą

„Pabaigos“ funkcija

2 dalis: įvedame duomenis (tekstą)

Poslinkis, kuriame
patalpinsime
išvedamą tekstą

```
-a 110
```

```
072A:0110 db 'Labas$'
```

```
072A:0116 db 0D
```

```
072A:0117 db 0A
```

```
072A:0118
```

Išvedamas tekstas
turi baigtis **\$**

Galima buvo
nerašyti :)

3 dalis: turime išsaugoti programą

Programos vardas

```
-n labas.com
```

```
-r CX
```

```
CX 0000 :18
```

```
-W
```

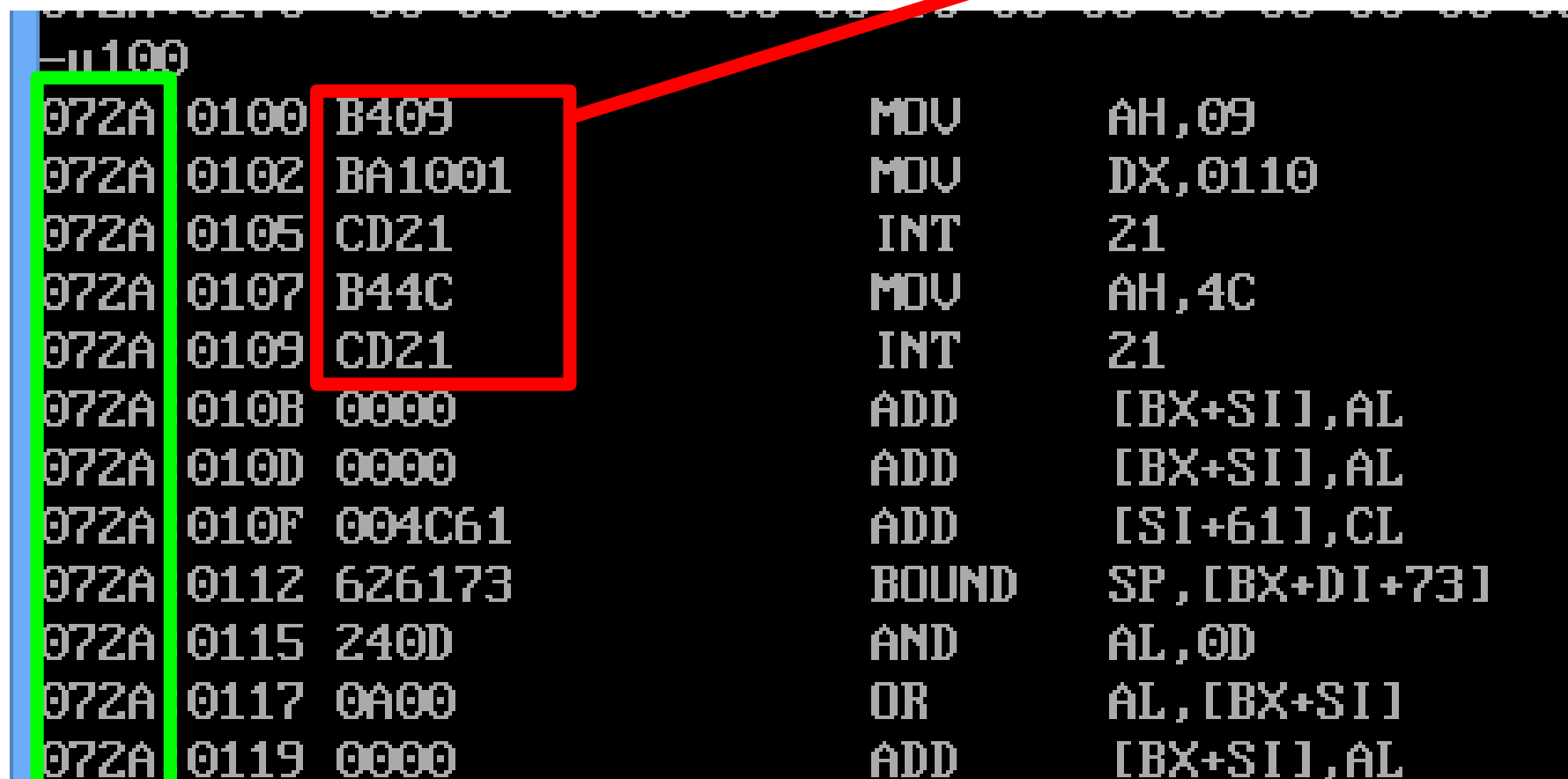
```
Writing 0010 bytes
```

Programos dydis
baitais

Įrašymo komanda

Gautos programos struktūra

Kodas



The diagram shows a memory dump with two annotations. A green box on the left highlights the segment register values (072A) and the offset values (0100 to 0119). A red box highlights the instruction codes (B409, BA1001, CD21, B44C, CD21). A green arrow points from the green box to the label 'CS' at the bottom. A red arrow points from the red box to the label 'Kodas' at the top right.

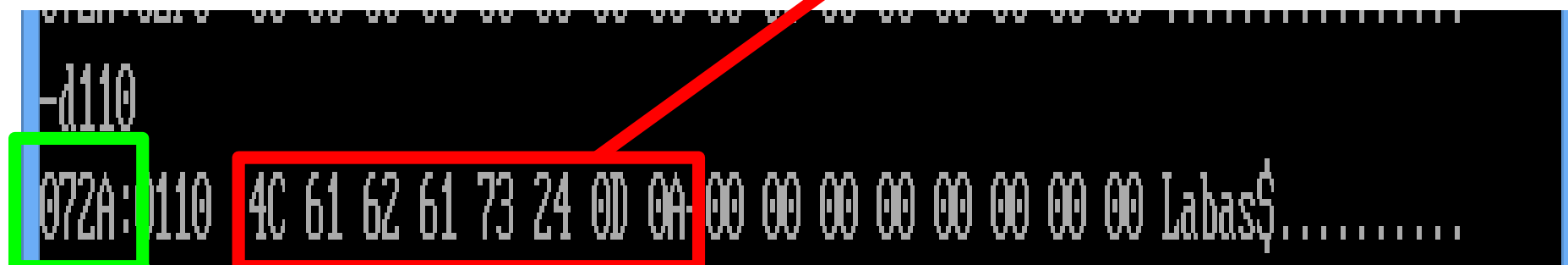
072A	0100	B409	MOV	AH,09
072A	0102	BA1001	MOV	DX,0110
072A	0105	CD21	INT	21
072A	0107	B44C	MOV	AH,4C
072A	0109	CD21	INT	21
072A	010B	0000	ADD	[BX+SI],AL
072A	010D	0000	ADD	[BX+SI],AL
072A	010F	004C61	ADD	[SI+61],CL
072A	0112	626173	BOUND	SP,[BX+DI+73]
072A	0115	240D	AND	AL,0D
072A	0117	0A00	OR	AL,[BX+SI]
072A	0119	0000	ADD	[BX+SI],AL

CS

KA-2

Gautos programos struktūra

Duomenys



DS

KA-2

Pastaba

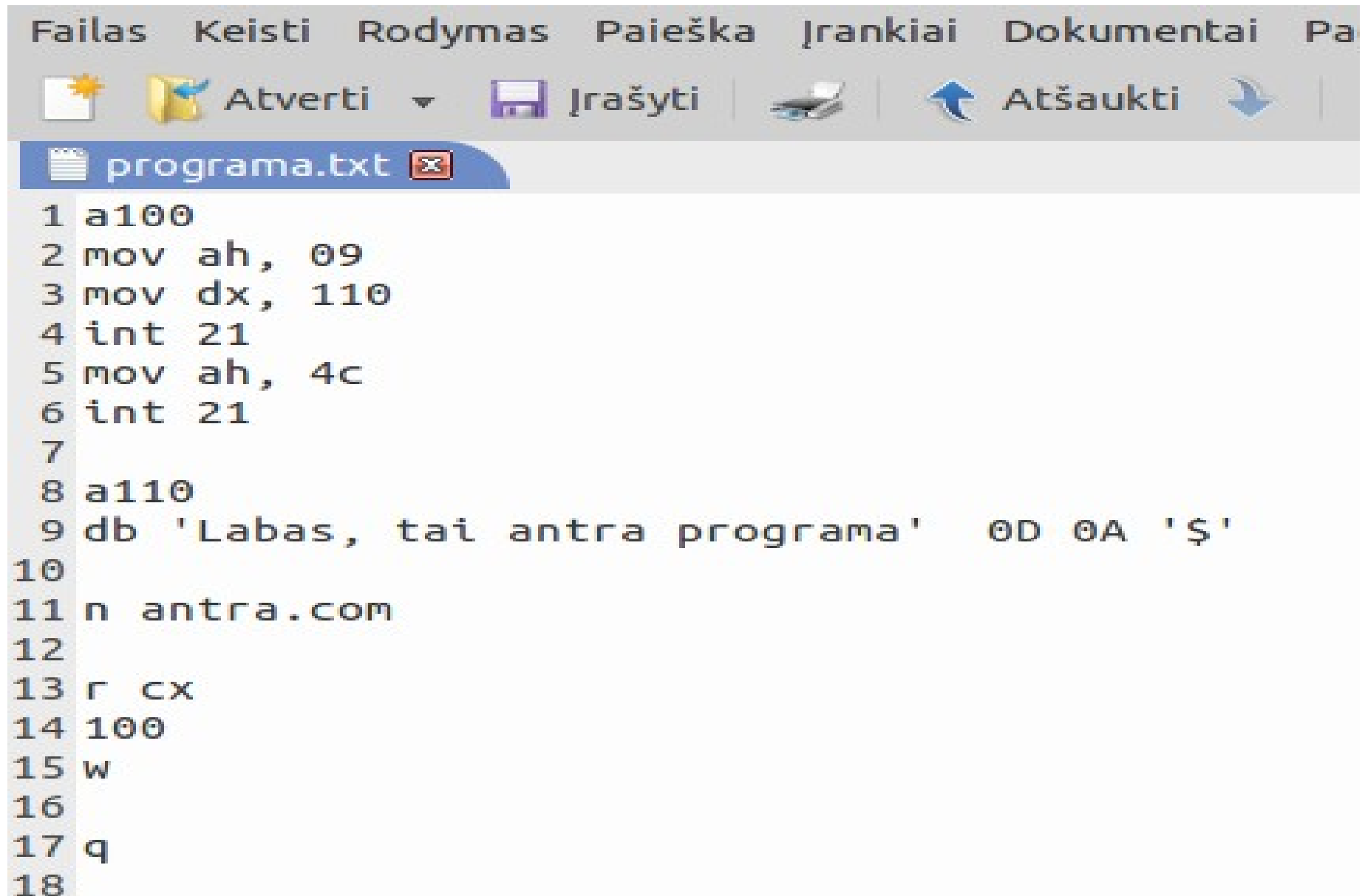
- Paleidus COM tipo programą CS,DS,ES ir SS sutampa.
 - ... tai nereiškia, kad vykdymo metu jų negalima keisti.

Kaip pagreitinti kodo rašymą

- Galima naudoti tekstinį failą, kuriame surašytos visos reikalingos komandos ir instrukcijos (žr. kitą skaidrę). Pvz., jeigu toks failas pavadintas programa.txt, tai galima paleisti jos debug su nukreipimu:

debug < programa.txt

Nukreipimas iš tekstinio failo



```
1 a100
2 mov ah, 09
3 mov dx, 110
4 int 21
5 mov ah, 4c
6 int 21
7
8 a110
9 db 'Labas, tai antra programa' 0D 0A '$'
10
11 n antra.com
12
13 r cx
14 100
15 w
16
17 q
18
```

Programos įvedimo pavyzdys

- Parašykime su **debug** programą, kuri spausdina žodį „Labas“. (Paaiškinimai – per pratybas)

```
a100  
mov dx, 200  
mov ah, 09  
int 21  
mov ah, 4c  
int 21
```

```
a200  
db 'Labas$'
```

```
n labas.com  
R CX  
200  
W  
q
```

Dar vienas pavyzdys

- Parašykime programą, kuri spausdina eilutę ABCDE, o po to – sukeičia joje raides C ir D vietomis ir vėl spausdina

```
a100
mov dx, 200
mov ah, 09
Int 21
mov ax, word ptr [202]
mov byte ptr [202], ah
mov byte ptr [203], al
mov dx, 200
mov ah, 09
Int 21
mov ah, 4c
Int 21
...
```

...kitas būdas

- ... panaudojame instrukciją XCHG

```
a100
mov dx, 200
mov ah, 09
Int 21
mov ax, word ptr [202]
xchg ah, al
mov word ptr [202], ax
mov dx, 200
mov ah, 09
Int 21
mov ah, 4c
Int 21
...
```

**Klausimas:
kiek baitų
sutaupėme?**

Atminties operandai

- Ankstesnėse programose panaudojome krepinius į atmintį. Yra bendra schema, pagal kurią formuojamas adresas atmintyje. **Iš kiekvieno stulpelio imame 0 arba 1 elementų (dviejų iš to paties stulpelio negalima).**

Poslinkis (16 bitų skaičius/ kai kuriais atvejais -8 bitų)	BX	SI
	BP	DI

KA-2

Atminties operandai

- Pavyzdžiai

```
mov ax, word ptr [1234]
```

```
...
```

```
mov cx, word ptr [1000 + bx]
```

```
...
```

```
mov cl, byte ptr [ ABCD + bx + di]
```

```
...
```

```
mov byte ptr [ ABCD + cx + di], al
```

```
...
```

```
mov byte ptr [1234][bx][si], al
```

```
...
```

```
mov byte ptr [1234] [di] [si], al
```

```
...
```

```
mov byte ptr [1234][bp][si], dl
```

```
...
```

```
mov byte ptr [1234][bx][si], 30
```

```
...
```

```
mov word ptr [1234][bx][si], 3031
```

Atminties operandai

- Pagal nutylėjimą adresas skaičiuojamas nuo DS pradžios, jeigu nenaudojamas **BP**. **BP atveju adresas skaičiuojamas SS atžvilgiu.**

nasm/yasm įvadas

- Nagrinėjame **yasmpvz1.asm**:
 - Programos struktūra
 - Bandome atlikti su debug

DOS funkcija 0A: teksto įvedimas iš klaviatūros

Kaip įvesti tekstą vykdomojoje programoje?

- Tam galima panaudoti 0A (dešimtą) DOS funkciją. Ji reikalauja įvesties **buferio**. Pirmas baitas tame buferyje rodo didžiausią simbolių skaičių, antras – faktiškai įvestų simbolių skaičius (be CR, t. y., 0D), nuo trečio baito pradedant – pati įvestoji eilutė (su CR).
- Pastaba. CR yra kursoriaus gražinimas į TOS PAČIOS eilutės pradžią :)

Pvz.: įveskime, „transformuokime“ ir
atspausdinkime tekstinę eilutę

Nagrinėjame
yasmpvz2.asm
koda

Dar vienas pavyzdys: nagrinėjame
yasmpvz3.asm

Individualioji programavimo užduotis

Nr. 1

Aptarimas

Pabaiga