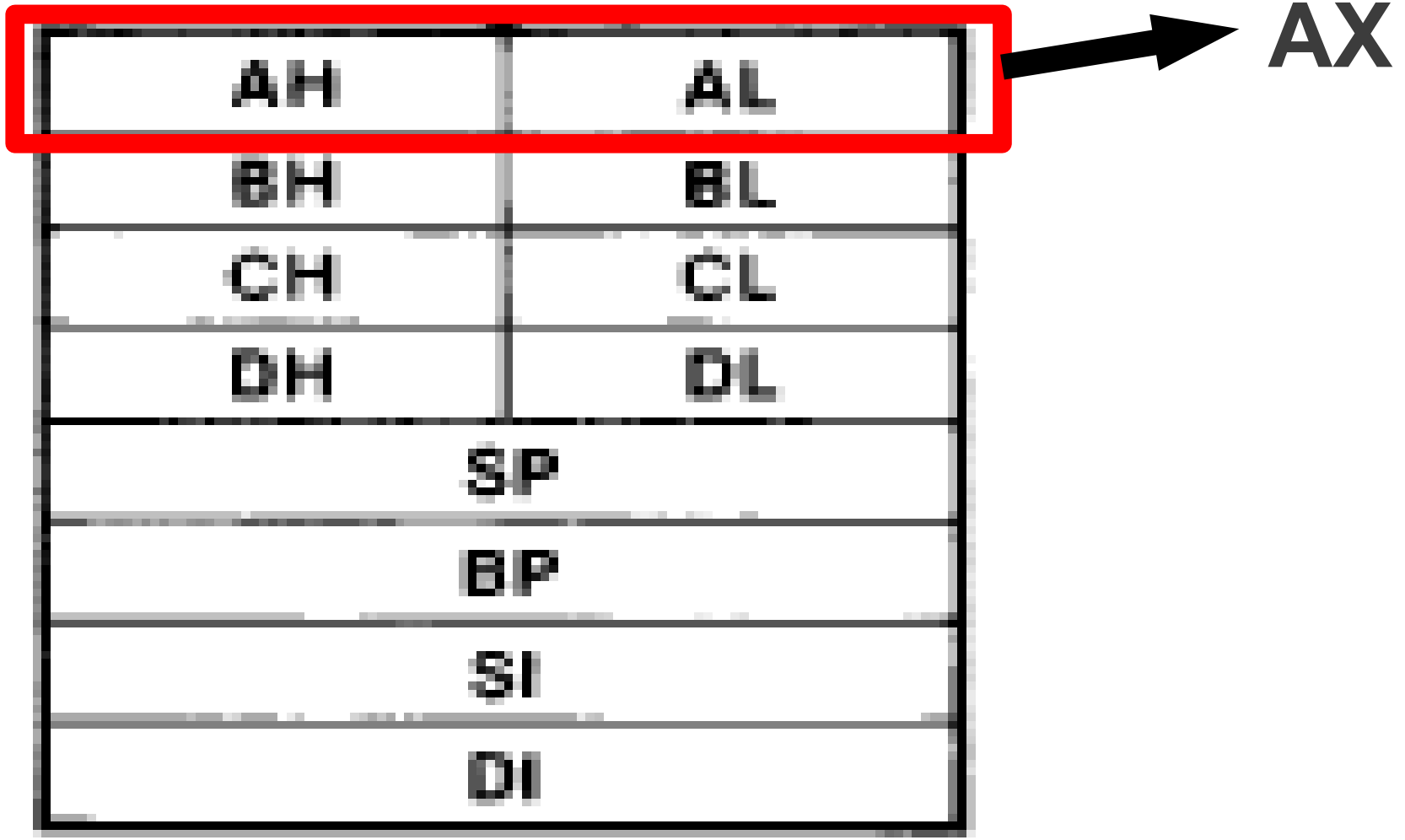
KA. Pratybos 2

I.Grinis

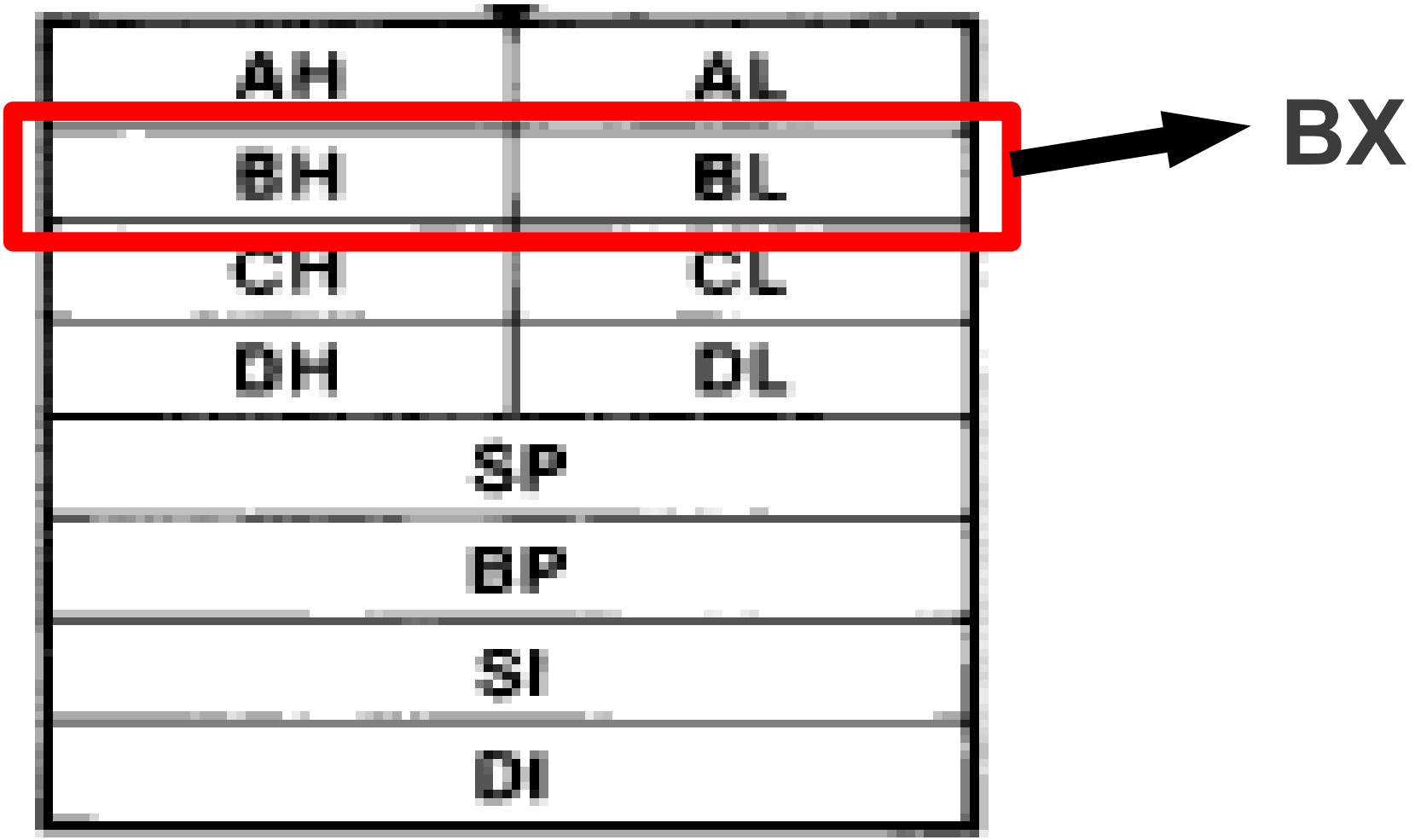
CPU sandara (registrai, jų kodas)

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	

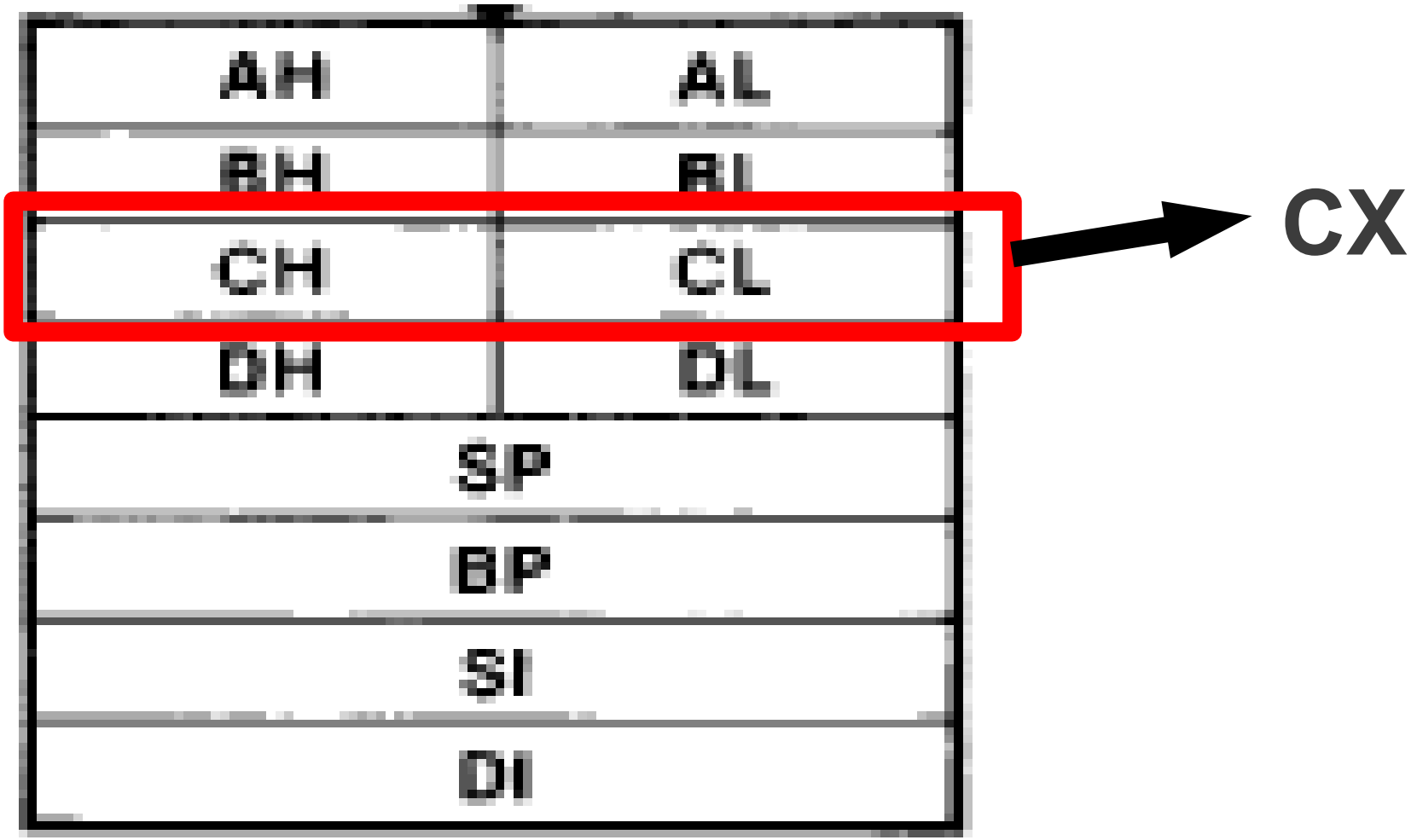
CPU sandara (bendrieji registrai)



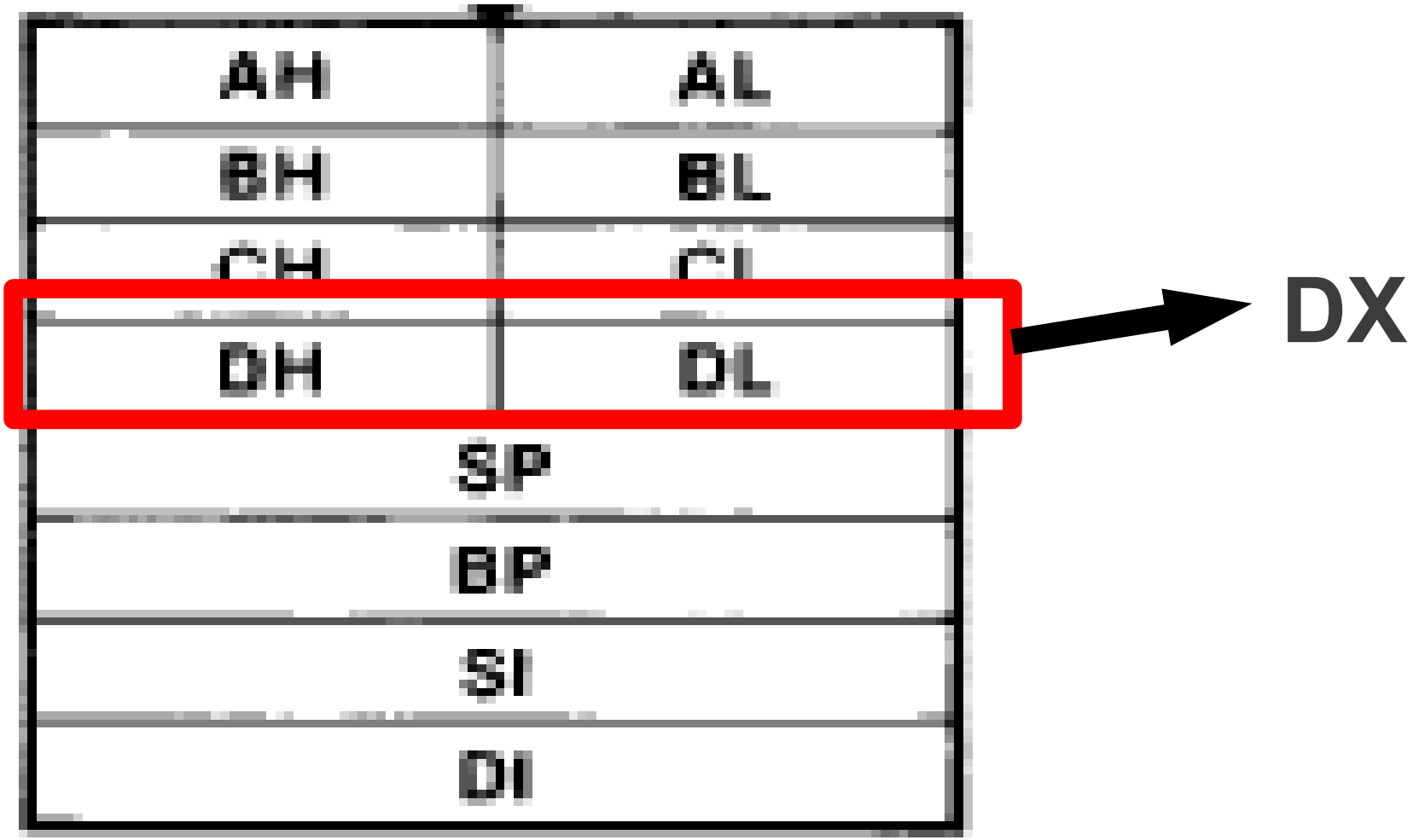
CPU sandara (bendrieji registrai)



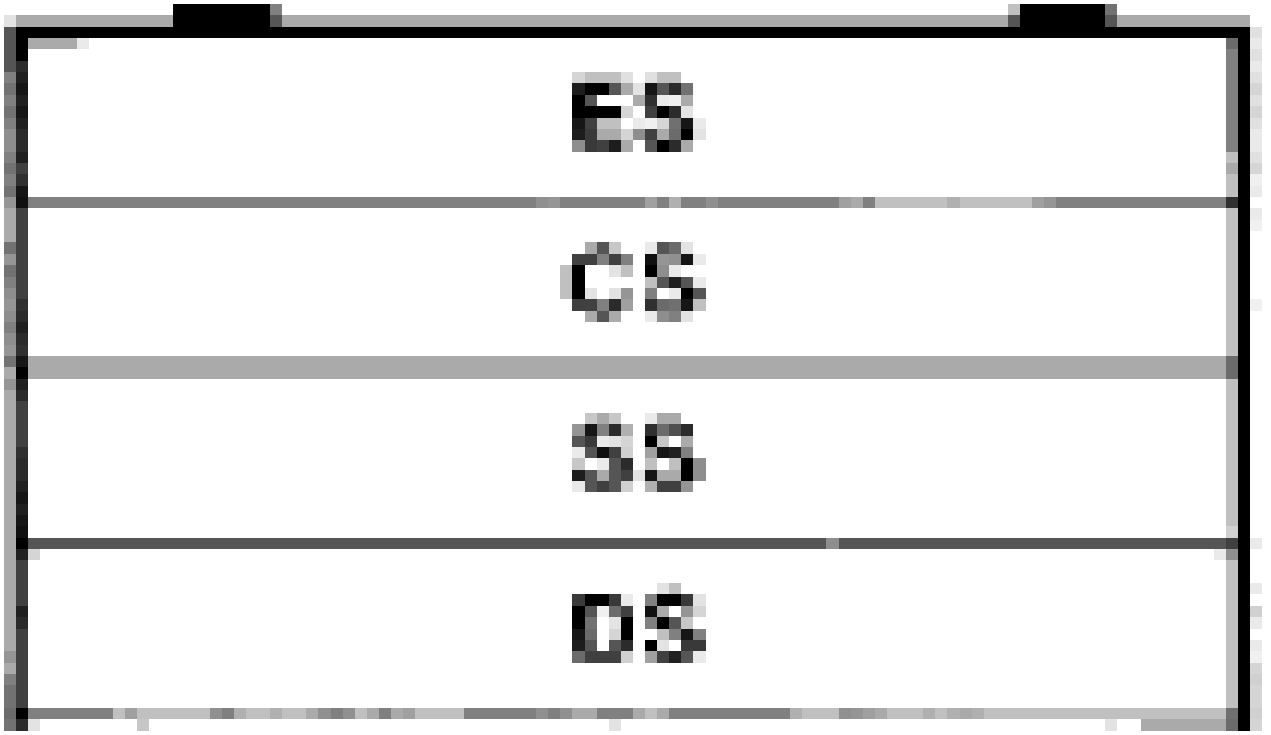
CPU sandara (registrai, jų kodas)



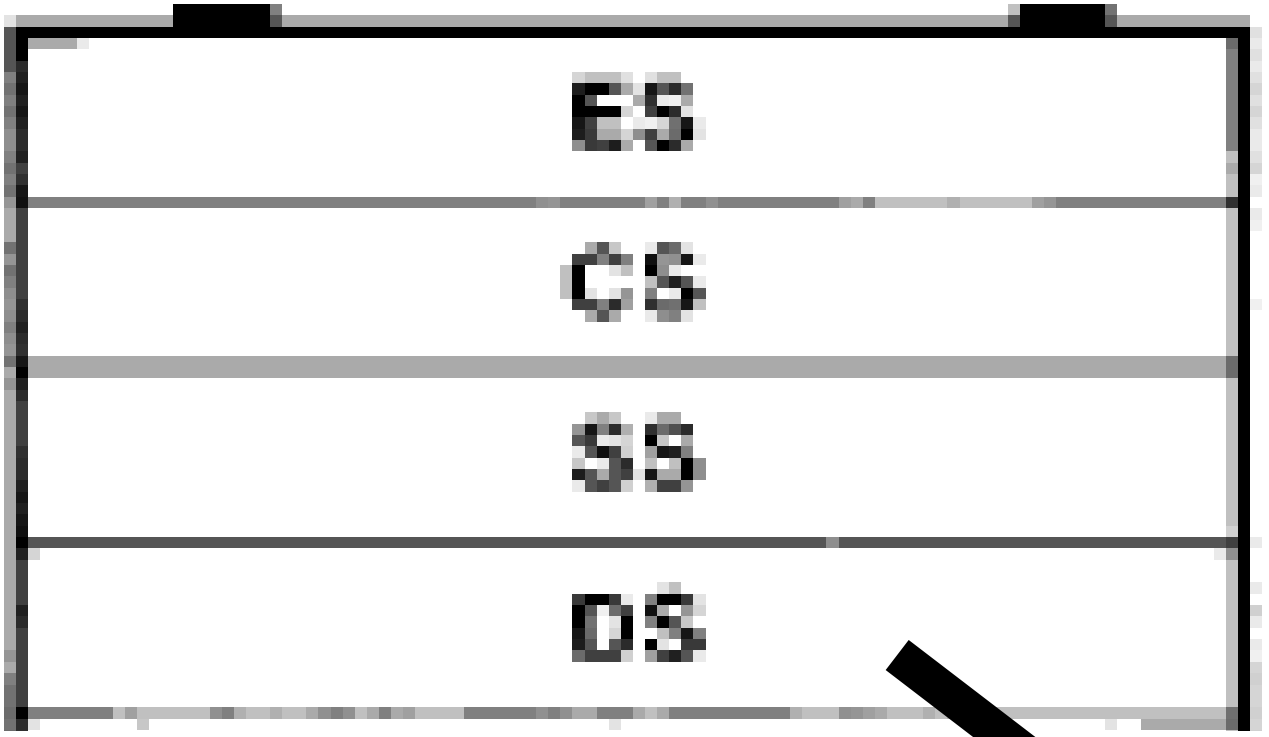
CPU sandara (registrai, jų kodas)



Segmentiniai registrai

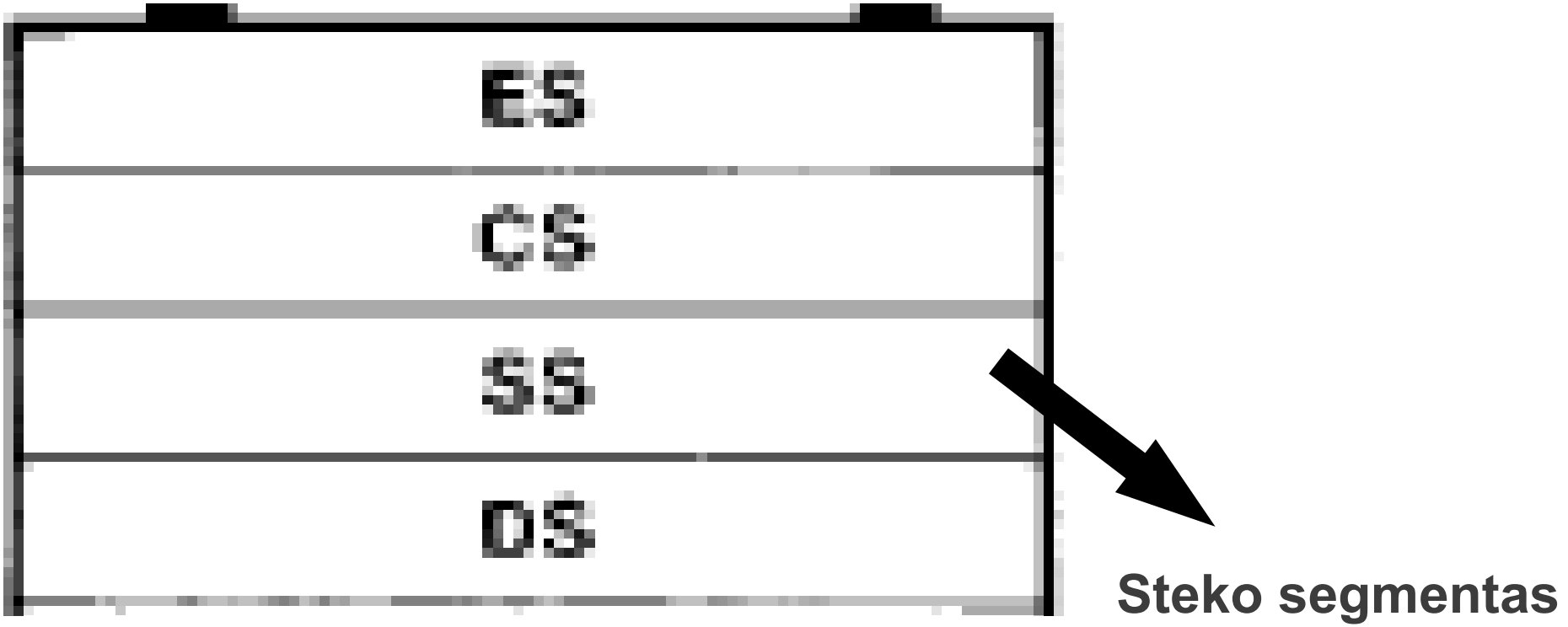


Segmentiniai registrai

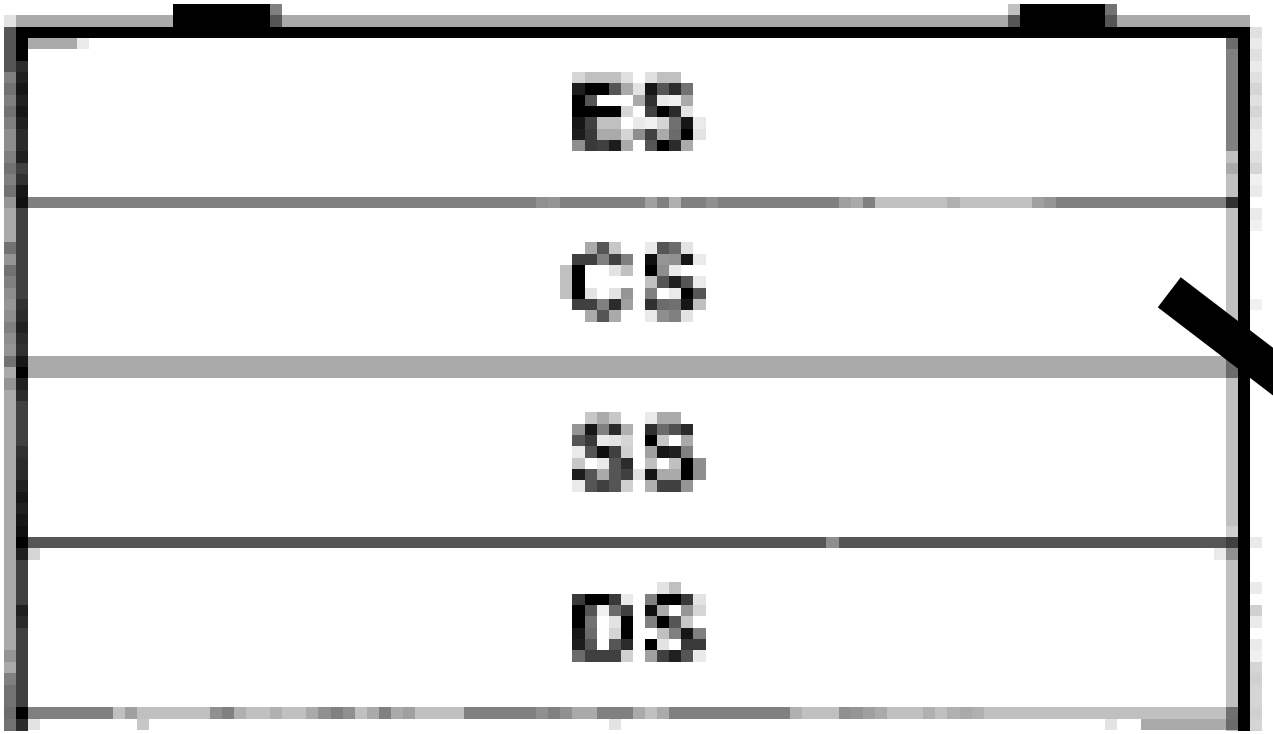


Duomenų segmentas

Segmentiniai registrai

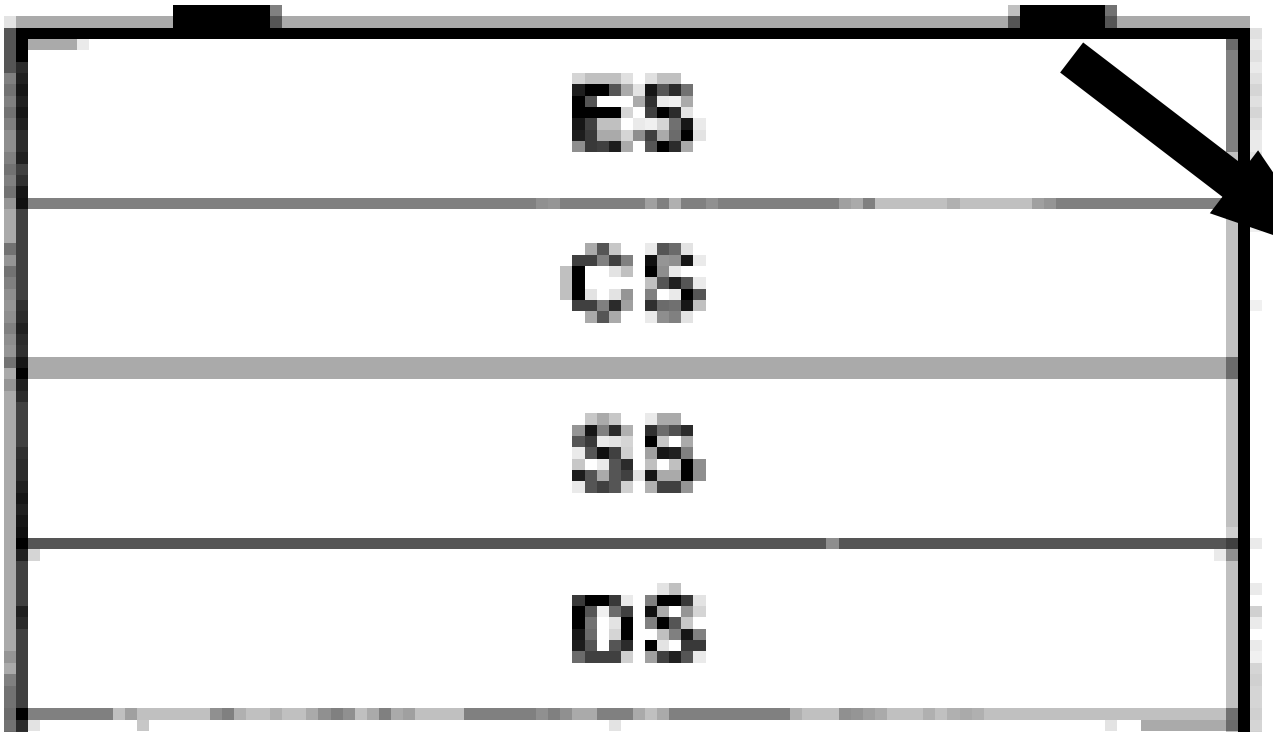


Segmentiniai registrai



Kodo segmentas

Segmentiniai registrai



Kaip formuojasi adresas?

- Klasikiniame PC gali būti iki 1 MB atminties, t. y., programuotojas turi gebėti adresuoti nuo 00000 iki FFFFF.
- 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.

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 = ...

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.
- Vykdant programas labai praverčia turėti specialią atminti, kurioje galima laikinai saugoti įvairią informaciją. Tam reikalingas stekas, kuris naudojamas vykdant tam tikras instrukcijas. Su steku susieti registrai yra SS (segmento) ir SP bei BP.

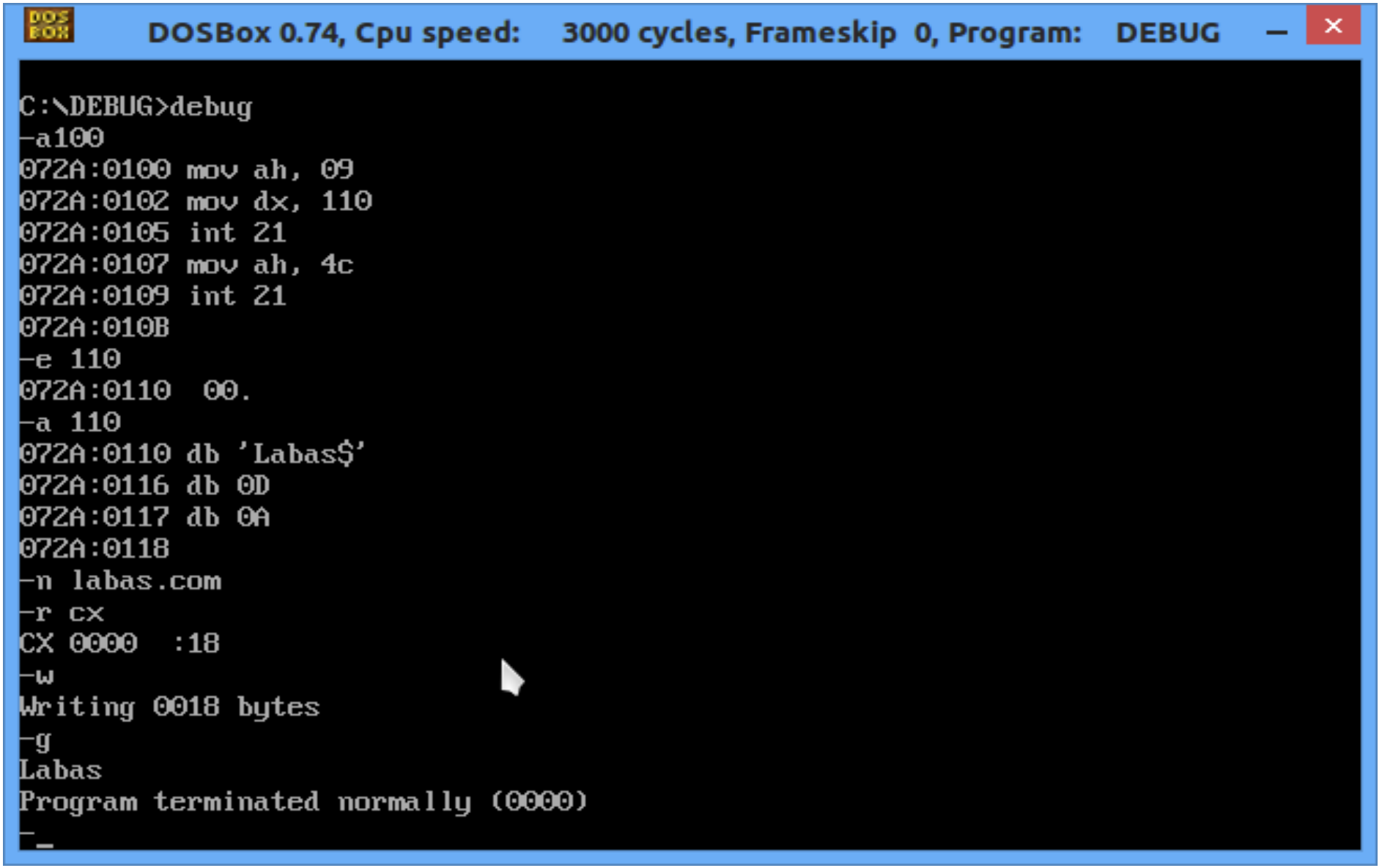
Debug programos nagrinėjimas

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

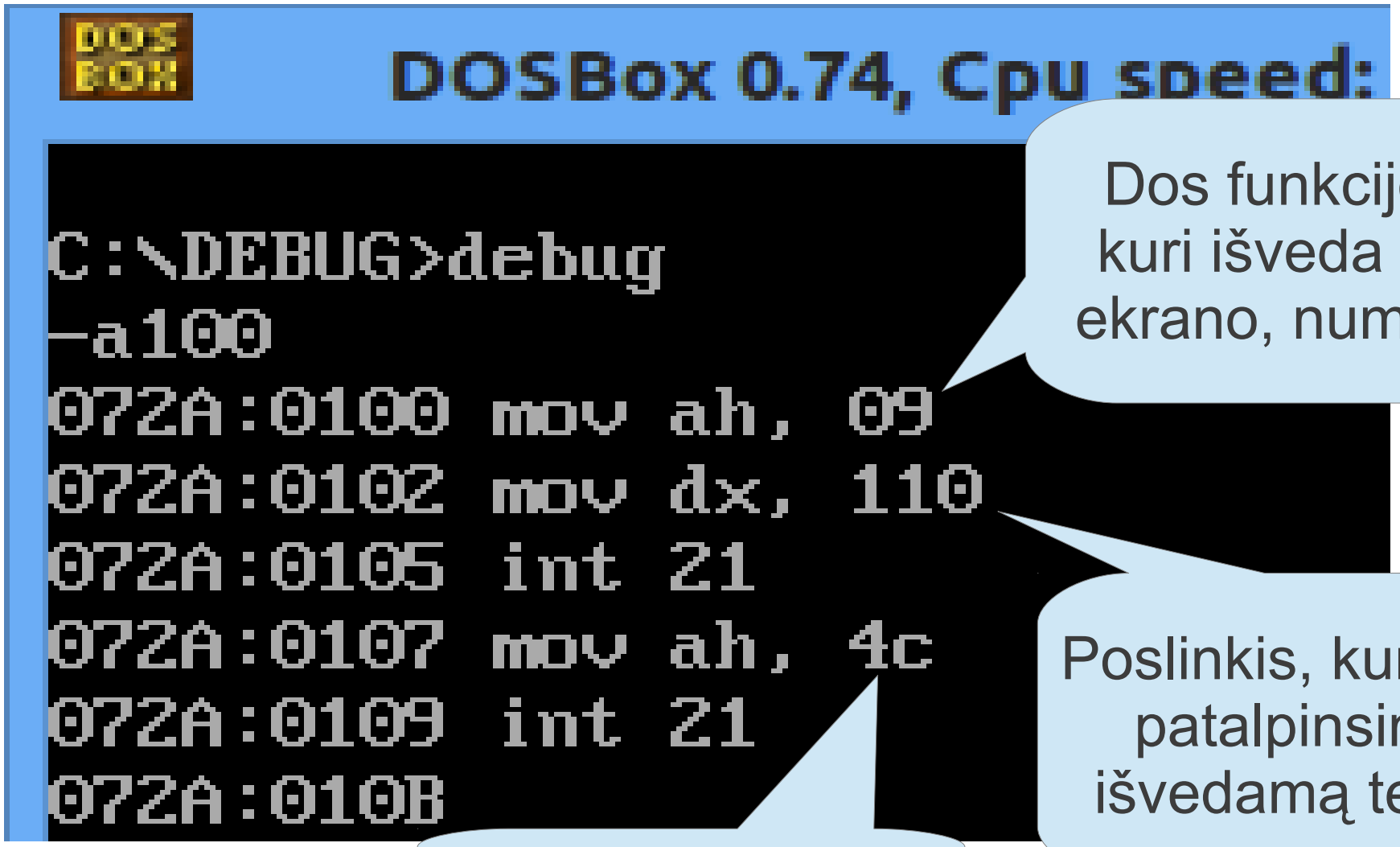
Debug panaudojimas kuriant paprastą programą

- 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.

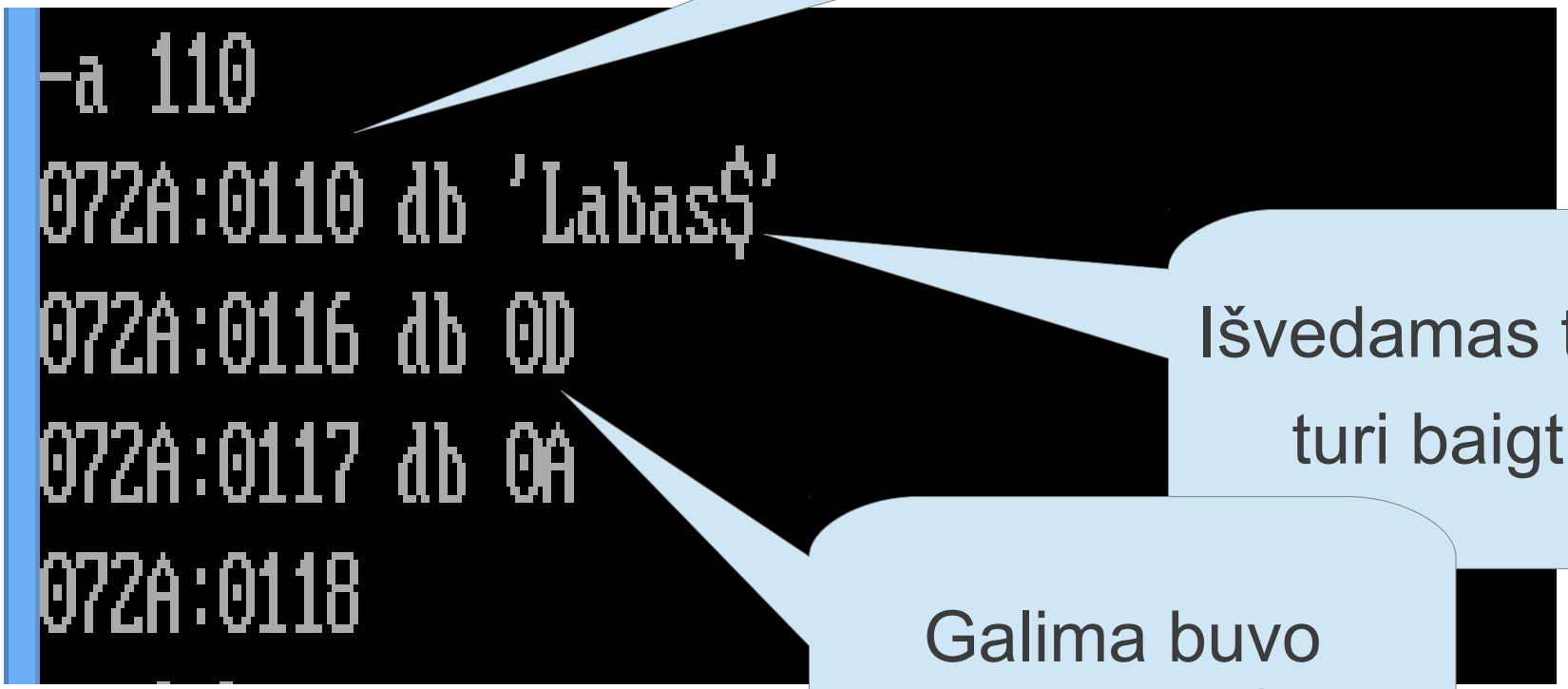
Programos įvedimo pavyzdys



1 dalis: įvedame kodą



2 dalis: įvedame duomenis (tekstą)

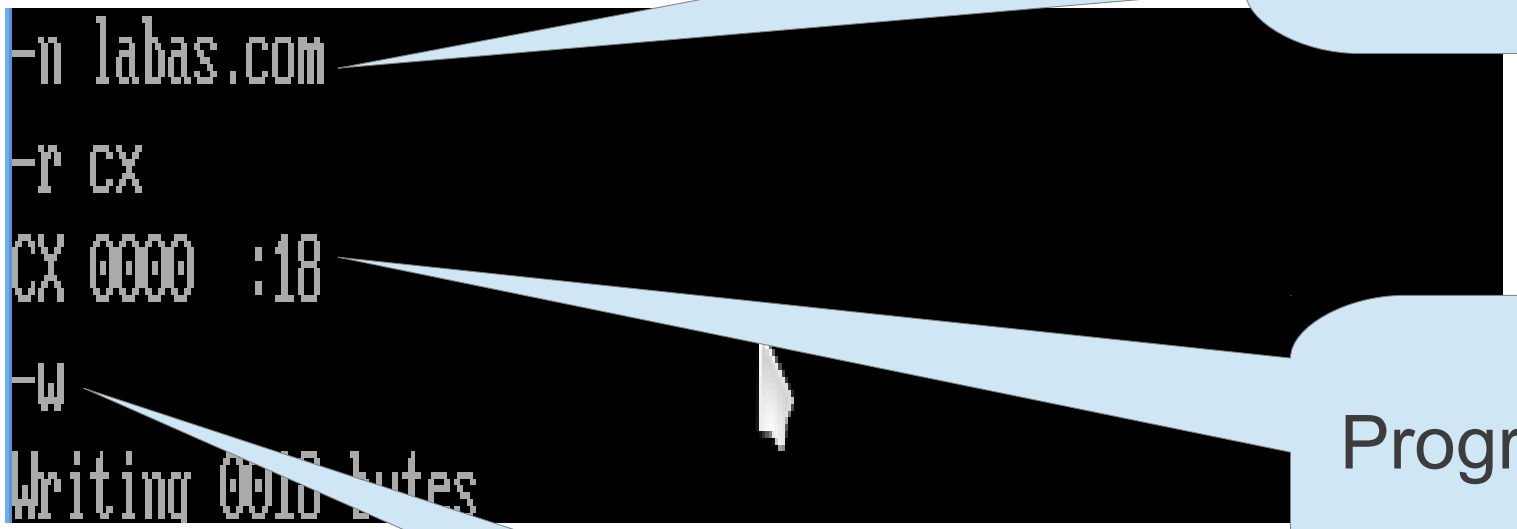


Poslinkis, kuriame patalpinsime išvedamą tekstą

Išvedamas tekstas turi baigtis **$**

Galima buvo nerašyti :)

3 dalis: turime išsaugoti programą



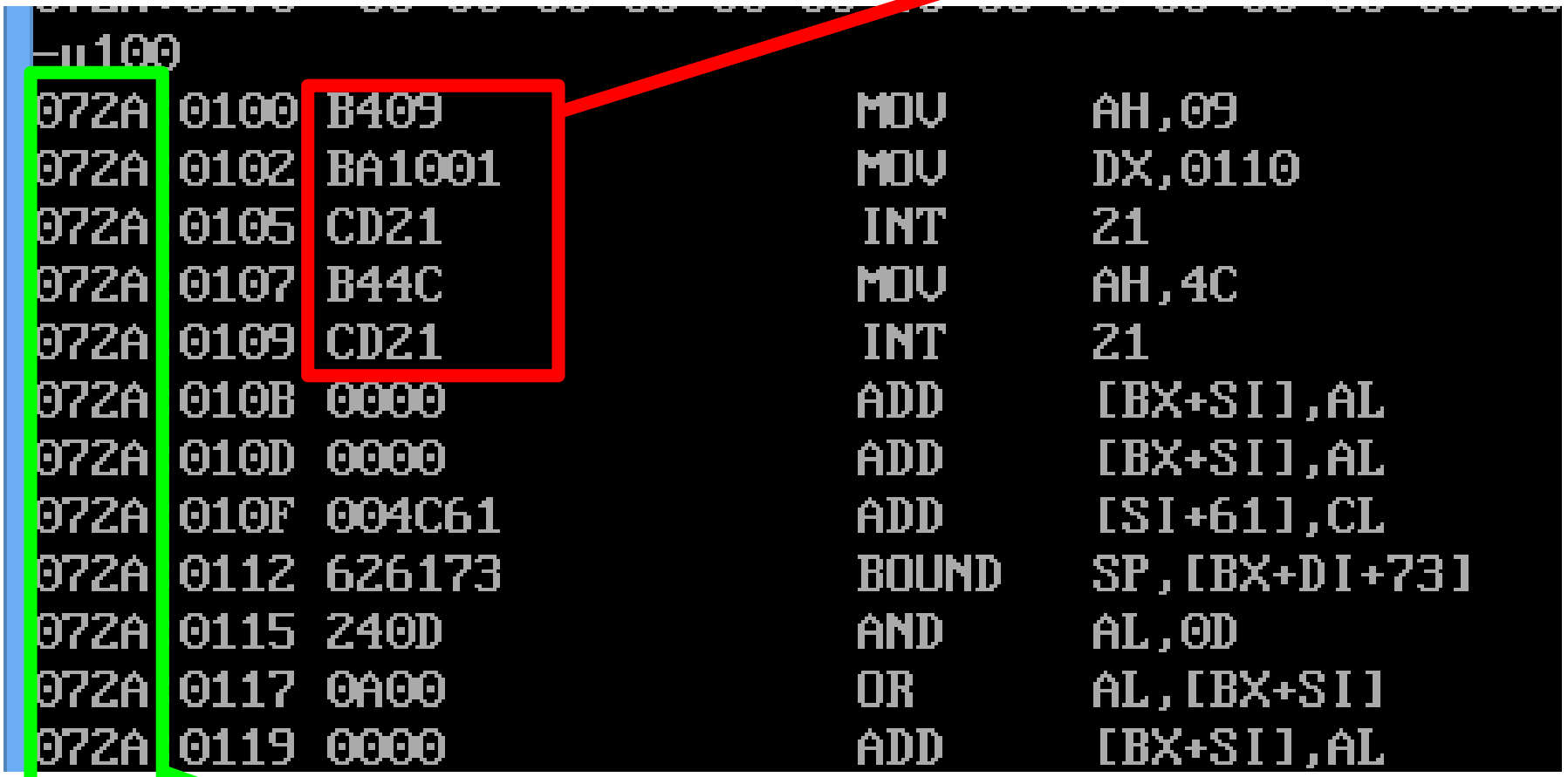
Programos vardas

Programos dydis baitais

Įrašymo komanda

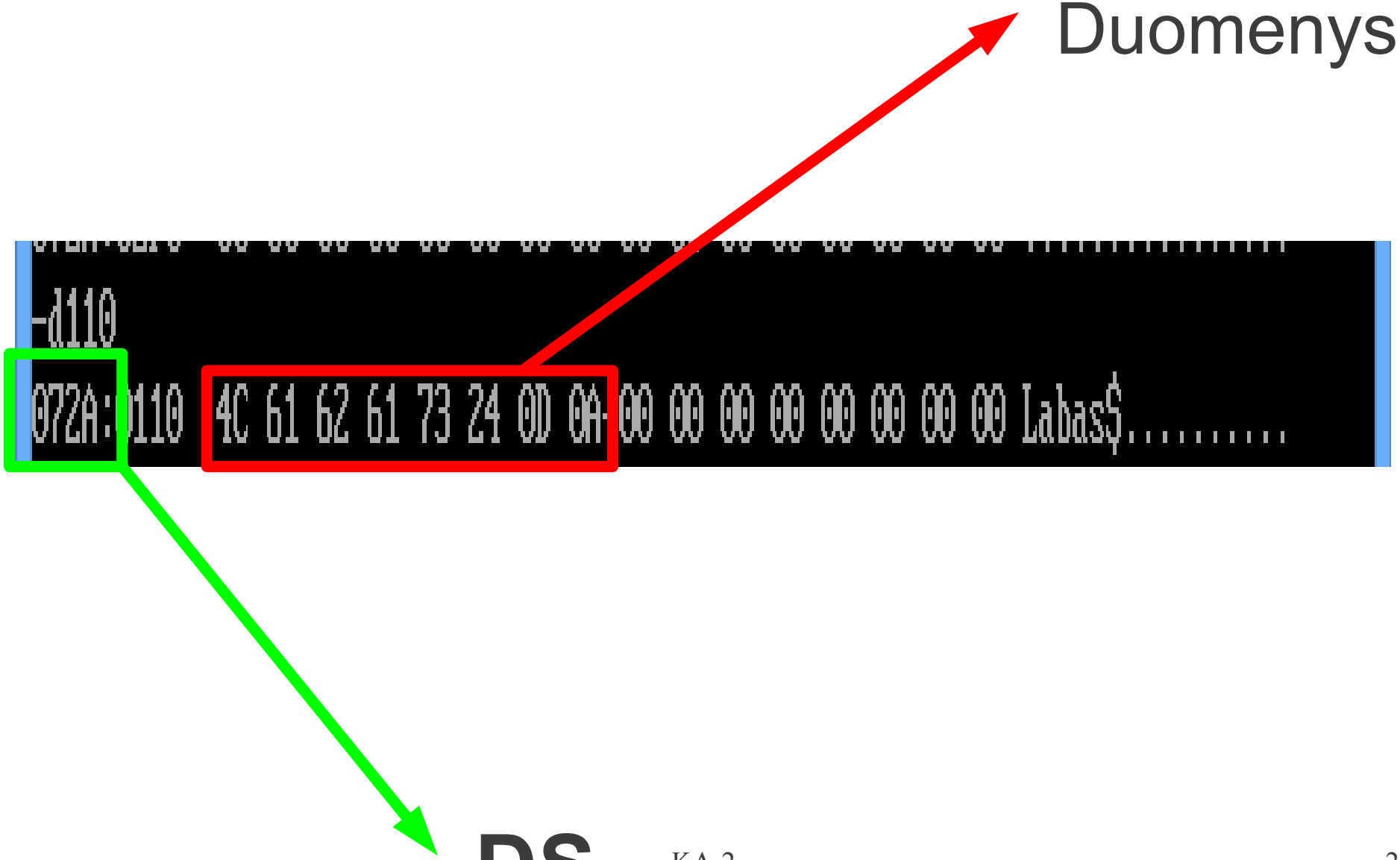
Gautos programos struktūra

Kodas



CS

Gautos programos struktūra



Pastaba

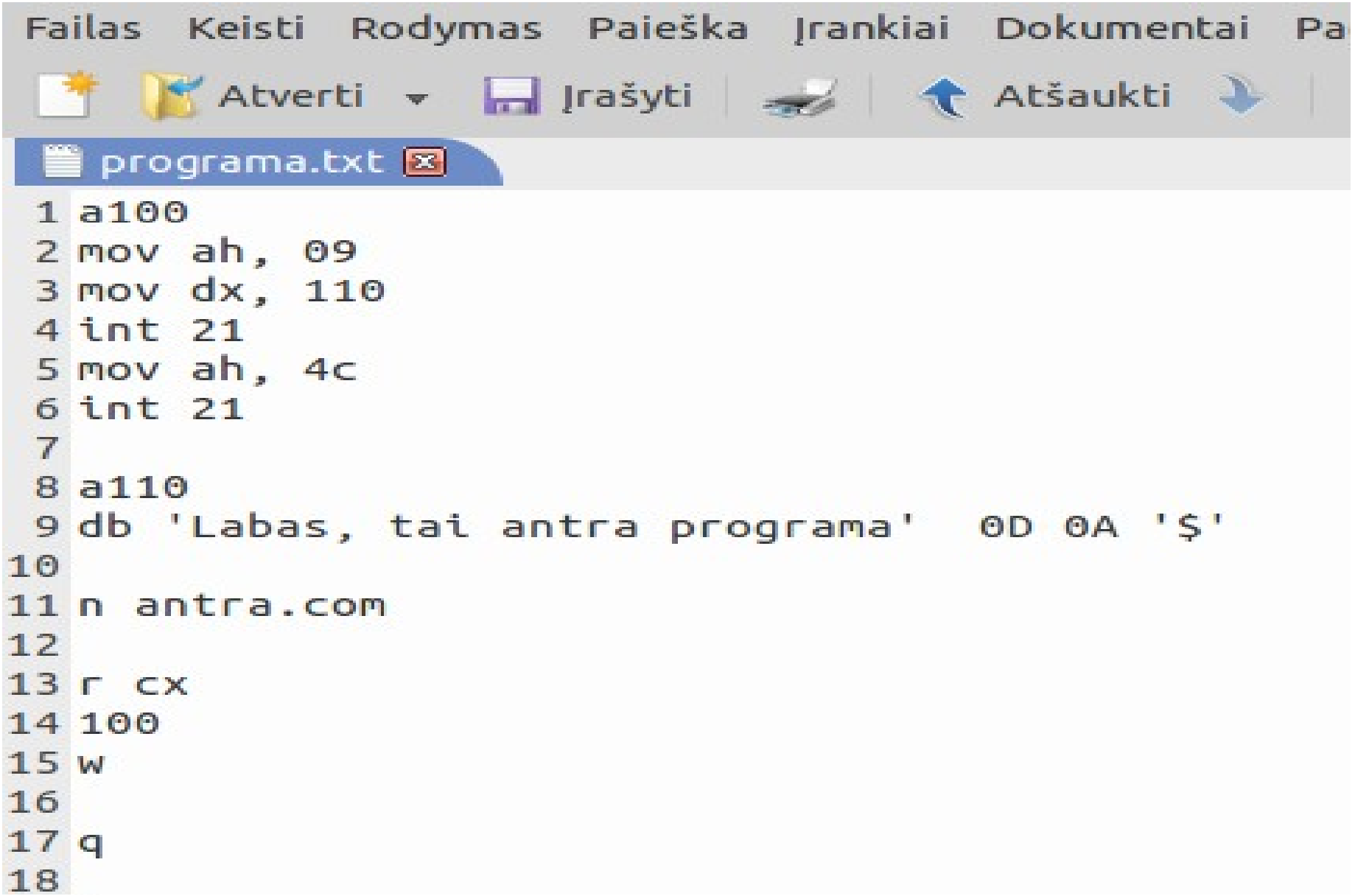
- Paleidus COM tipo programą CS,DS,ES ir SS sutampa.
  - … tai nereiškia, kad vykdymo metu jų negalima keisti.

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



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**

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.**

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

Žr. programų „ivesk.asm“ ir „iveskv2.asm“ kodą

Individualioji užduotis Nr. 1 (0,2 balo, dvi savaitės)

- **Parašykite pagal analogiją su „ivesk.asm“ programą, kurioje vartotojas įveda iki 8 simbolių eilutę (arba nurodyto fiksuoto ilgio) ir padaro jos „transformaciją“ pagal Jums nurodytą būdą ir išveda rezultatą ant ekrano.**
  - **Pastaba. Jums turėtų užtekti tik gero MOV instrukcijos suvokimo ir gebėjimų ja naudotis**

Variantai A

1) Eilutės paskutinis simbolis sukeičiamas su pirmuoju, o trečias pakeičiamas tarpu;

2) Eilutės paskutinis simbolis sukeičiamas su priešpaskutiniuoju, o antras pakeičiamas tarpu;

3) Eilutės paskutinis simbolis sukeičiamas su antruoju, o trečias pakeičiamas '*';

4) Eilutės paskutinis simbolis sukeičiamas su priešpaskutiniuoju, o antras pakeičiamas '+';

Variantai A

5) Eilutės paskutinis simbolis pakeičiamas 'A', o trečias - tarpu, o pirmas ir antras – sukeičiami vietomis;

6) Eilutės priešpaskutinis simbolis pakeičiamas 'P', o antras - sukeičiamas su paskutiniuoju;

7) Eilutės paskutinis simbolis pakeičiamas trečiuoju, o pats trečias pakeičiamas '*';

8) Eilutės paskutinis simbolis sukeičiamas su priešpaskutiniuoju, o antras pakeičiamas 'Z';

Variantai B

9) Eilutės ilgis visada yra 6. Padaroma transformacija pagal schemą: ABCDEF → ABECD

10) Eilutės ilgis visada yra 5. Padaroma transformacija pagal schemą: ABCDE → AABBECCDD

11) Eilutės ilgis visada yra 6. Padaroma transformacija pagal schemą: ABCDEF → CBDEFA

12) Eilutės ilgis visada yra 6. Padaroma transformacija pagal schemą: ABCDEF → BDFACE

13) Eilutės ilgis visada yra 8. Padaroma transformacija pagal schemą: ABCDEFGH → ABGHCDEF

CDEF

14) Eilutės ilgis visada yra 3. Padaroma transformacija pagal schemą: ABC → ABCCBAACB

Variantai B

15) Eilutės ilgis visada yra 3. Padaroma transformacija pagal schemą: ABC → A B C C B A (atsiranda tarpai)

16) Eilutės ilgis visada yra 5. Padaroma transformacija pagal schemą: ABCDE → AB CD EE (atsiranda tarpai)

17) Eilutės ilgis visada yra 6. Padaroma transformacija pagal schemą: ABCDEF → DE FA CC (atsiranda tarpai)

18) Eilutės ilgis visada yra 6. Padaroma transformacija pagal schemą: ABCDEF → A C E B D F (atsiranda tarpai)

19) Eilutės ilgis visada yra 4. Padaroma transformacija pagal schemą: ABCD → A B D C D (atsiranda tarpai)

20) Eilutės ilgis visada yra 3. Padaroma transformacija pagal schemą: ABC → A,B,C,C,B (atsiranda kableliai)

Pabaiga