Mitašiūnas Programming Language

Skaidres ir medžiagą ruošė:
Jonas Brusokas
Benediktas <Benas> Gricius

Vilniaus universiteto
Studentų atstovybė

mifsa

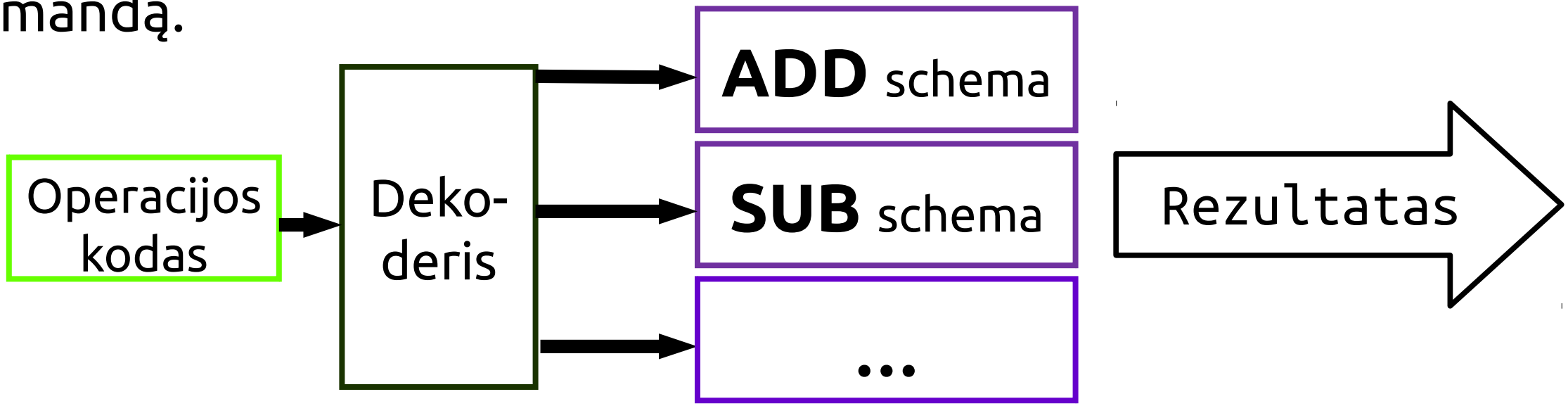
Ką mes jau žinom

Mes žinome, kad modernia programavimo kalba parašytas kodas yra paverčiamas į assemblerinę mnemoniką ir tada iš assemblerinio kodo į mašininį kodą (kartais iš karto į mašininį kodą). Tačiau kaip procesorius žino ką daryti su mašininiu kodu?

```
X = X + 69;      ➡  ADD varX, 069h
ADD varX, 069h   ➡  FF 8A 69
FF 8A 69         ➡  ?
```

Anksčiau...

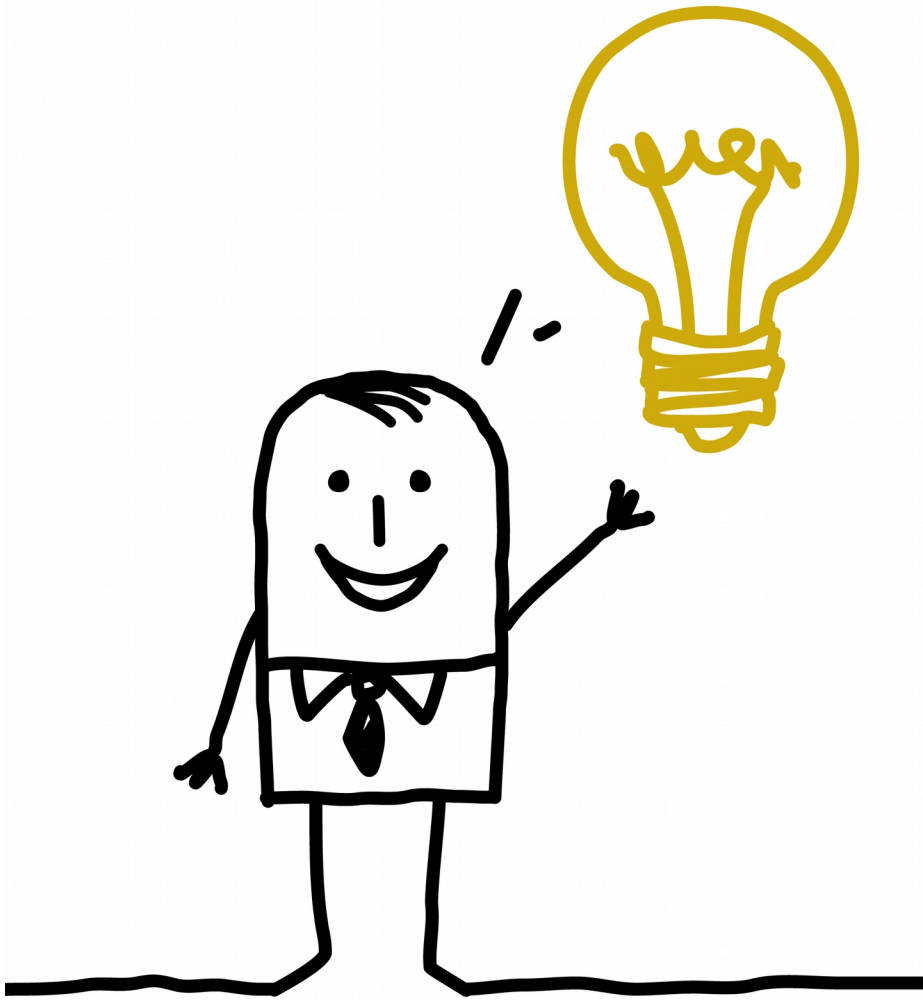
Pirmose architektūrose kiekviena komanda turėjo po atskirą „schemą“, kuri realizuodavo konkrečią standartinę assembly komandą.



Šita metodika buvo gera tol, kol architektūros buvo primityvios, tačiau padaugėjus komandų kiekiui kilo poreikis pereiti prie apdorojimo metodo.

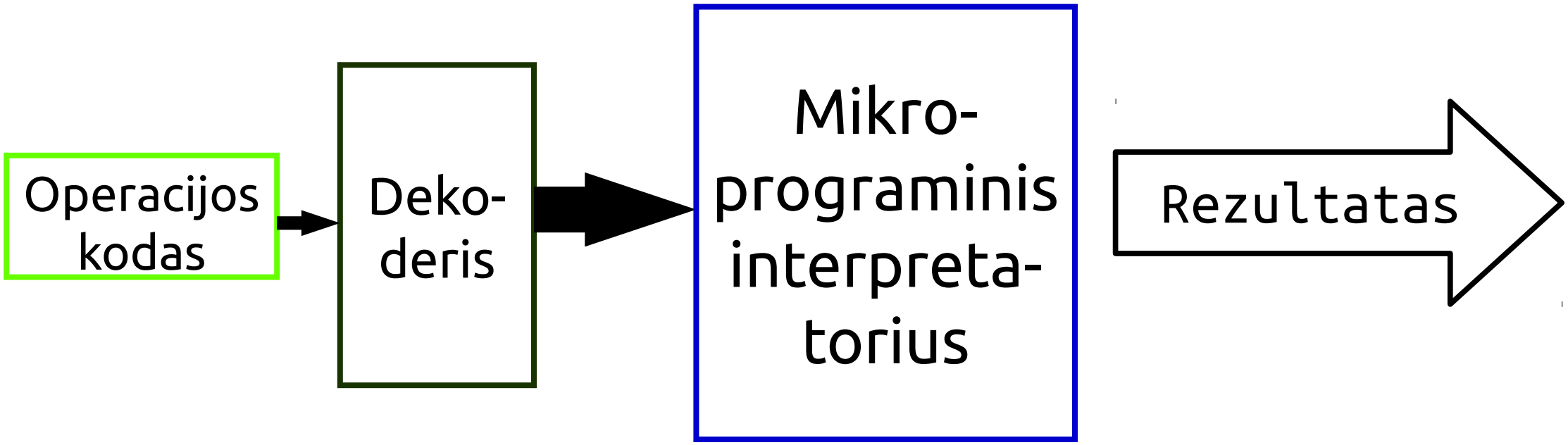
Buvo pastebėta, kad skirtingos kompleksinės komandos buvo **realizuojamos primityviomis operacijomis**.

Kilo mintis sukurti vieną **unifikuotą schemą**, kurią sudarytų visa naudojamų primityvių **operacijų aibė**, ir **jai sukurti instrukcijų sekas**, kurios atitiktų kiekvieną norimą realizuoti komandą.



... Dabar

Dabar komandos yra realizuojamos naudojant schemos instrukcijas (***Mikrokomandas***). Kiekviena komanda aparatūriniame lygmenyje yra realizuojama atskira ***mikrokomandų*** seka (***Mikroprograma***).



Mikroprogramavimas

Mikroprogramavimas yra procesoriaus komandų realizavimas mikromandomis aparatūrinio lygio schemose.

Mikrokomanda yra aparatūrinės interpretacijos schemos instrukcija, kurios veikimas yra tiesiogiai paremtas fiziniu schemos pavidalu ir apribojimais. (jei schemoje negalima, mikrokomandos taip pat nėra)

Mikroprogramavimui (komandų realizavimu mikrokomandomis) yra naudojama **MPL**, žemiausio įmanomo lygio programavimo kalba.

Java

Microsoft Visual C#

python

Perl

COBOL PROGRAMMING

C++

C Language

PASCAL

Assembly Programming Language

MPL

MPL

MPL – *Microprocessor programming language*

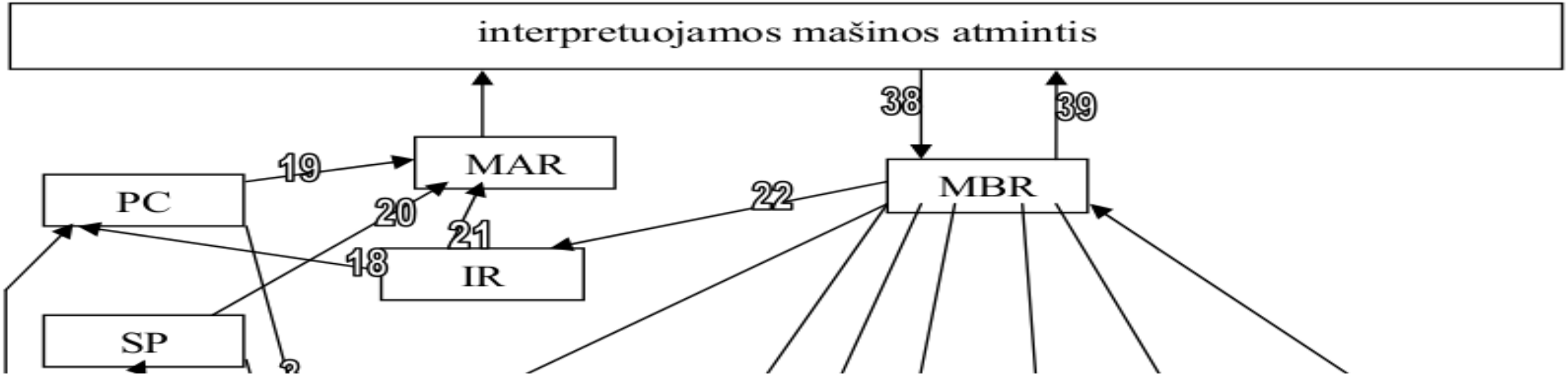
- Giliau už asemblerį
- ASM žinios nebūtinos temai suprasti, bet praverstų
- Atspindi realų ASM įgyvendinimą aparatūroje (hardware'e)

P.S. ASM – asembleris, you knew it already...

Didn't you? ;)

Interpretuojamos mašinos interpretatorius

```
0       MAINLOOP: MAR = PC; MBR = MEMORY(MAR);
1                 IR = MBR; PC = PC + 1;
2                 IF BIT(15, IR) = 1 THEN GOTO OP4567;
3                 IF BIT(14, IR) = 1 THEN GOTO OP23;
4                 IF BIT(13, IR) = 1 THEN GOTO POP;
5       PUSH:     MAR = IR; SP = SP + 1; MBR = MEMORY(MAR);
6                 MAR = SP; MEMORY(MAR) = MBR;
7                 GOTO MAINLOOP;
8       POP:      MAR = SP; SP = SP + (-1); MBR = MEMORY(MAR);
9                 MAR = IR; MEMORY(MAR) = MBR;
10                GOTO MAINLOOP;
```



Nauja eilutė – nauja komanda.

- **Viena komanda:**

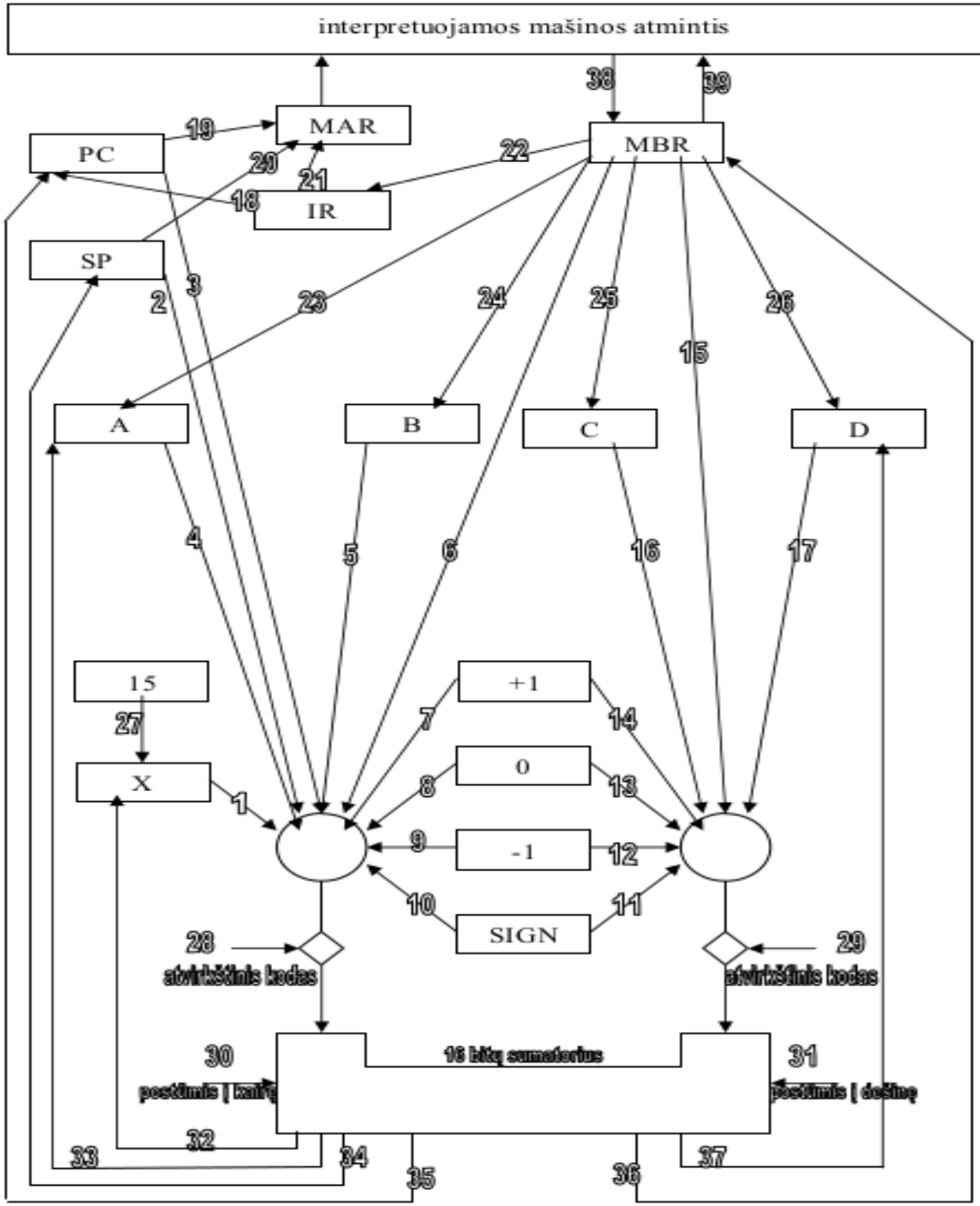
  X=15; MBR=1+1;

- **Dvi komandos:**

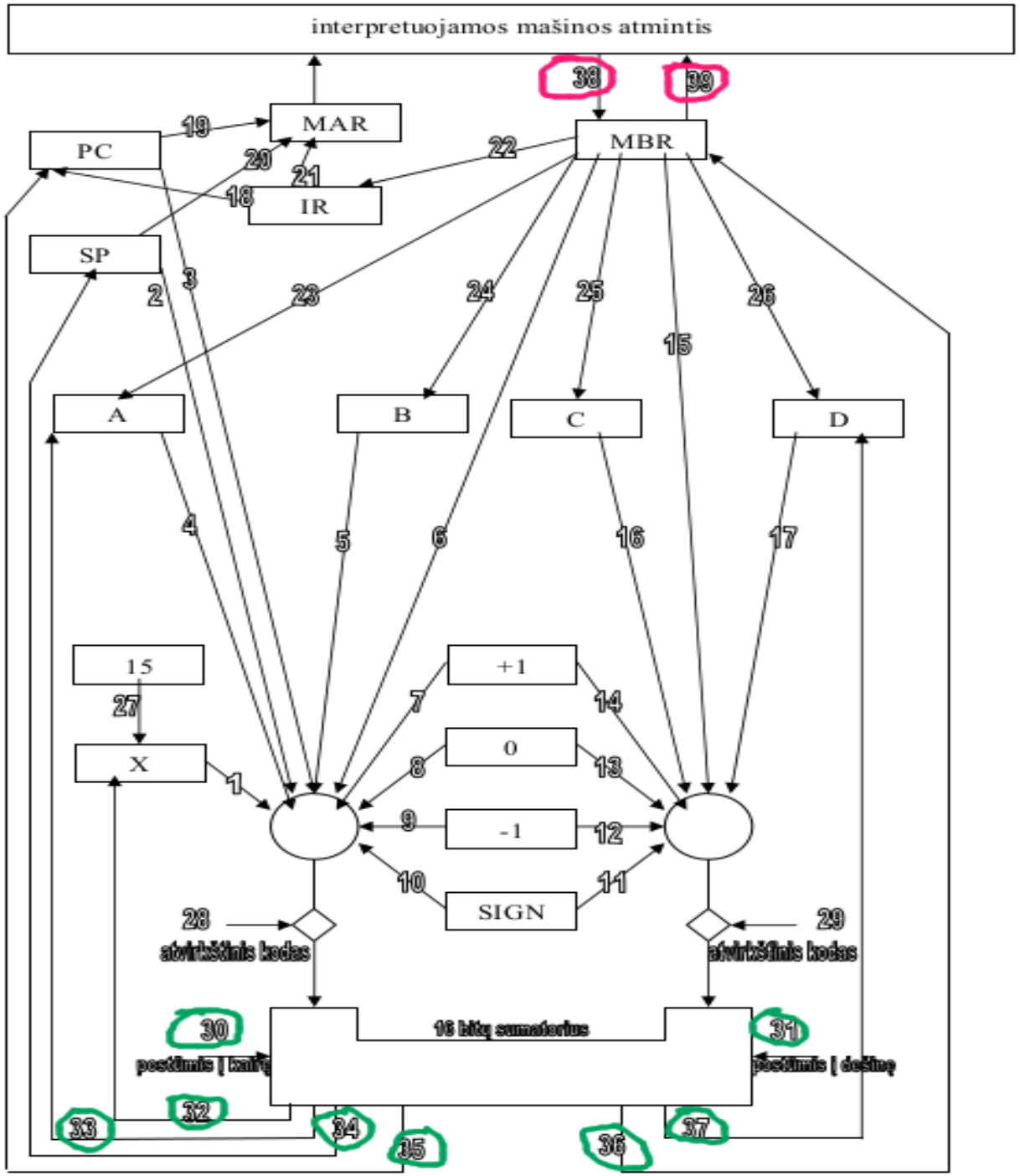
  X=15;

  MBR=1+1;

  Viena komanda reiškia paraleliai vykdomus veiksmus, nekorektiška suformuoti komandas, kurios grąžina neapibrėžtus rezultatus!


interpretuojamos mašinos atmintis
MAR
MBR
PC
IR
SP
A
B
C
D
15
+1
0
-1
SIGN
X
28
29
atvirkščiasis kodas
atvirkščiasis kodas
30
31
16 bitų sumatorius
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
32
33
34
35
36
37
38
39

16psl.



Pocikliai:

1 – nepažymėtas

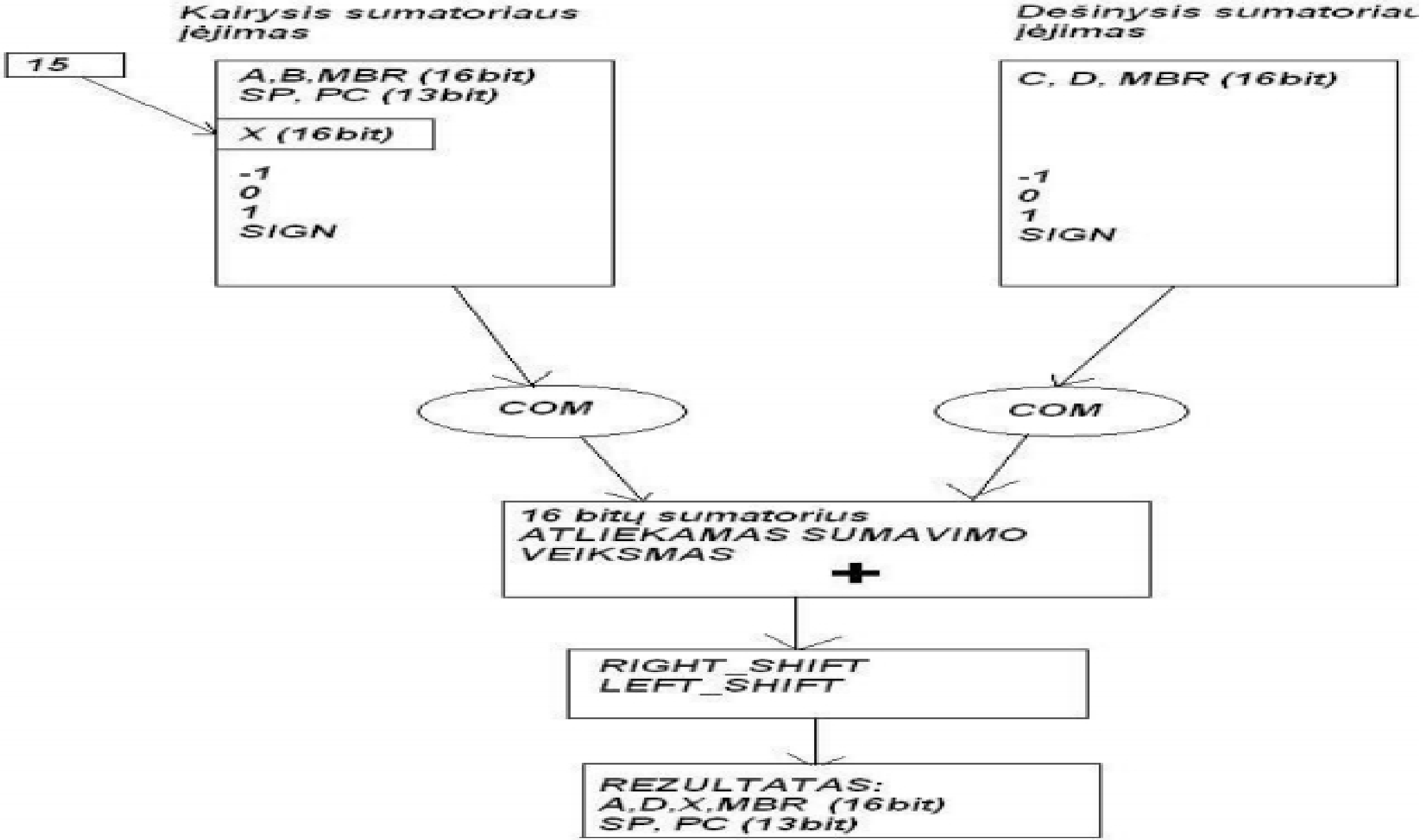
2 – žaliai

3 – purple

Būtina išvengti neapibrėžtumo

Beware - Race condition!

NEGALIMA: **X**=15; A=**X**+1;


Kairysis sumatoriaus įėjimas
Dešinysis sumatoriaus įėjimas
15
A,B,MBR (16bit)
SP, PC (13bit)
X (16bit)
-1
0
1
SIGN
C, D, MBR (16bit)
-1
0
1
SIGN
COM
COM
16 bitų sumatorius
ATLIEKAMAS SUMAVIMO VEIKSMAS
+
RIGHT_SHIFT
LEFT_SHIFT
REZULTATAS:
A,D,X,MBR (16bit)
SP, PC (13bit)

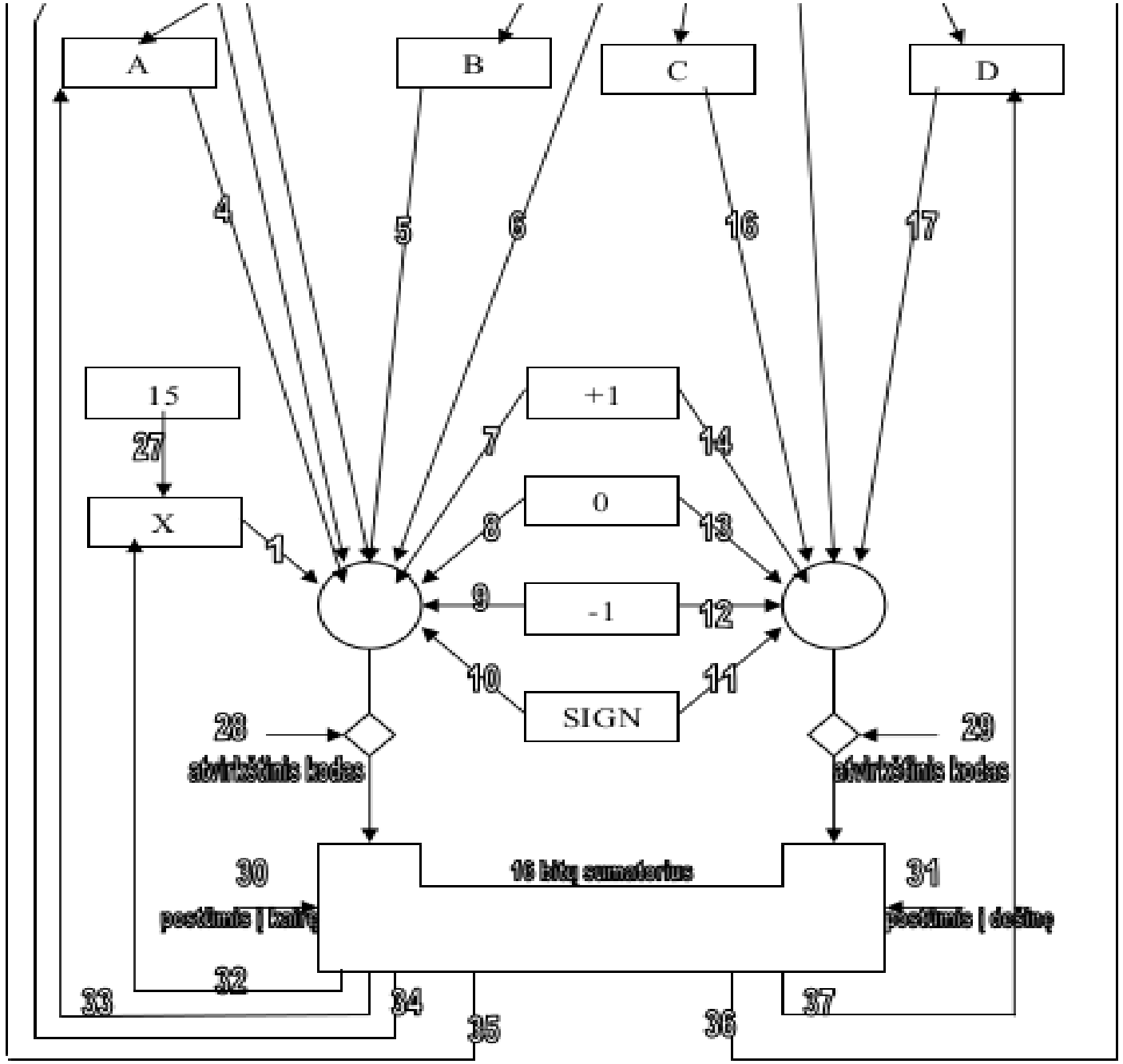
NEW SLIDE: Normalu abiejose pusėse turėti tą patį

- Jei abiejose pusėse yra tas pasirinkimas, t.y. galima:

  MBR = 1+1;

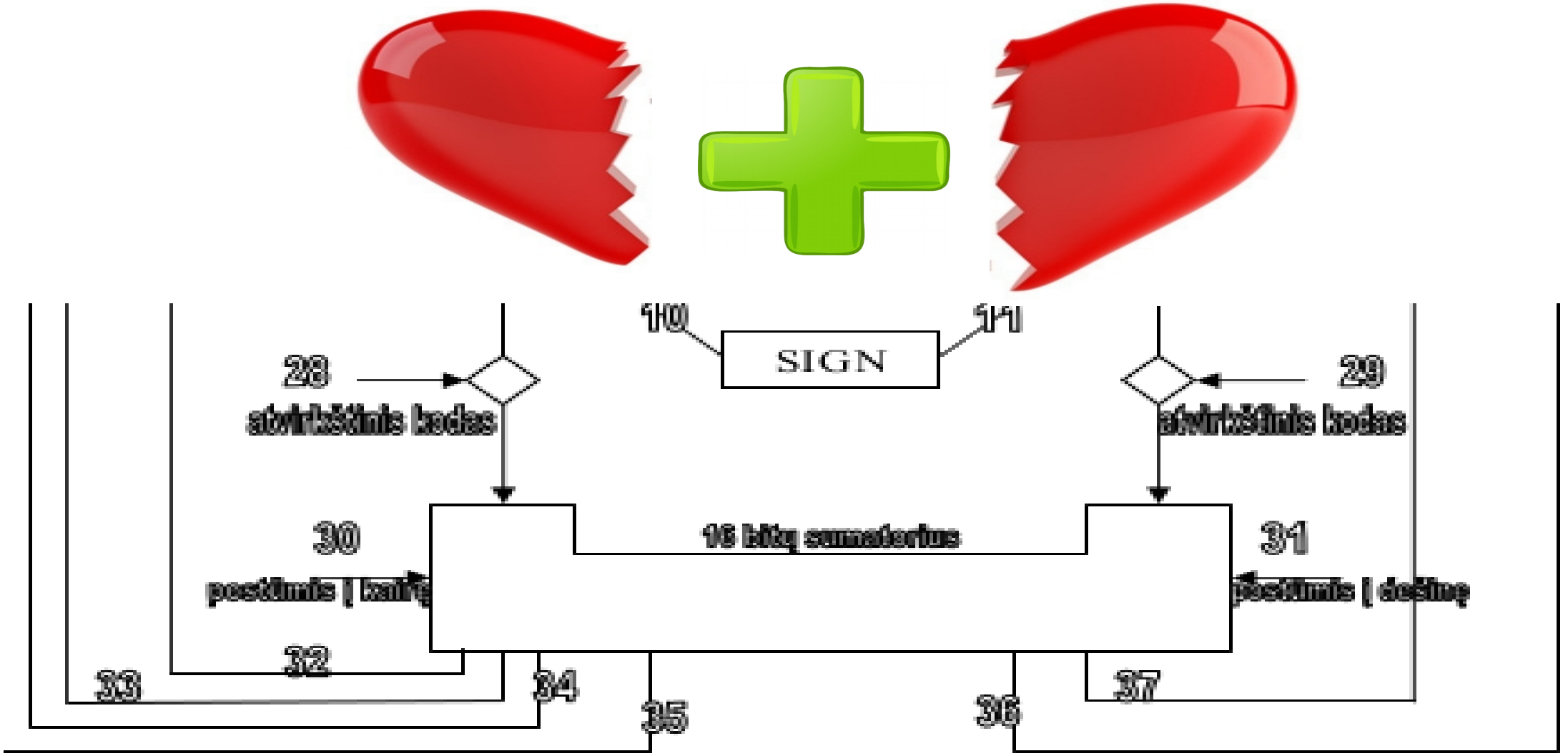
  MBR = MBR + MBR;

  bet **negalima**:
  MBR = A + A; (hint: *į kurią pusę A ateina iš tikrųjų?*)

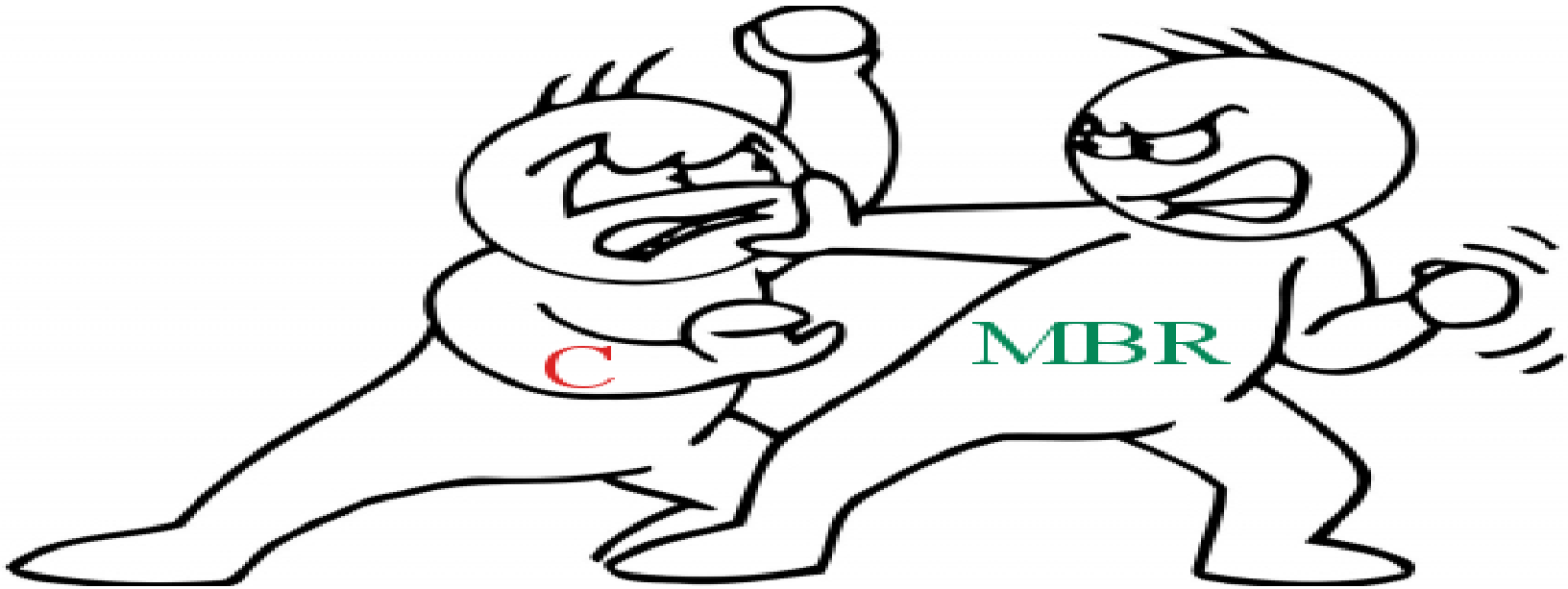


Moka tik sudėti

Į abi puses reikia kažką pasiųsti



Ne daugiau kaip po vieną kiekvienoje pusėje



… bet ir ne mažiau (nedaužykite širdžių)

Noriu pasakyt, kad...

- **Negalima**: MBR=1;

- **Vis tiek reikia:**

  MBR = 1 + 0;

  arba

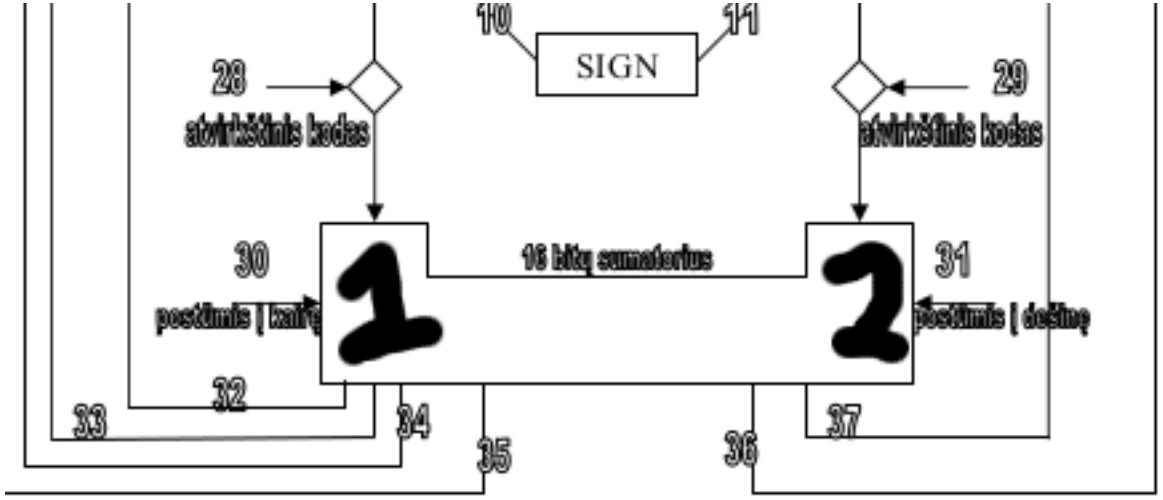
  MBR = 0 + 1;

- **Tačiau galima:** A=A+1; (hint: *pocikliai*)

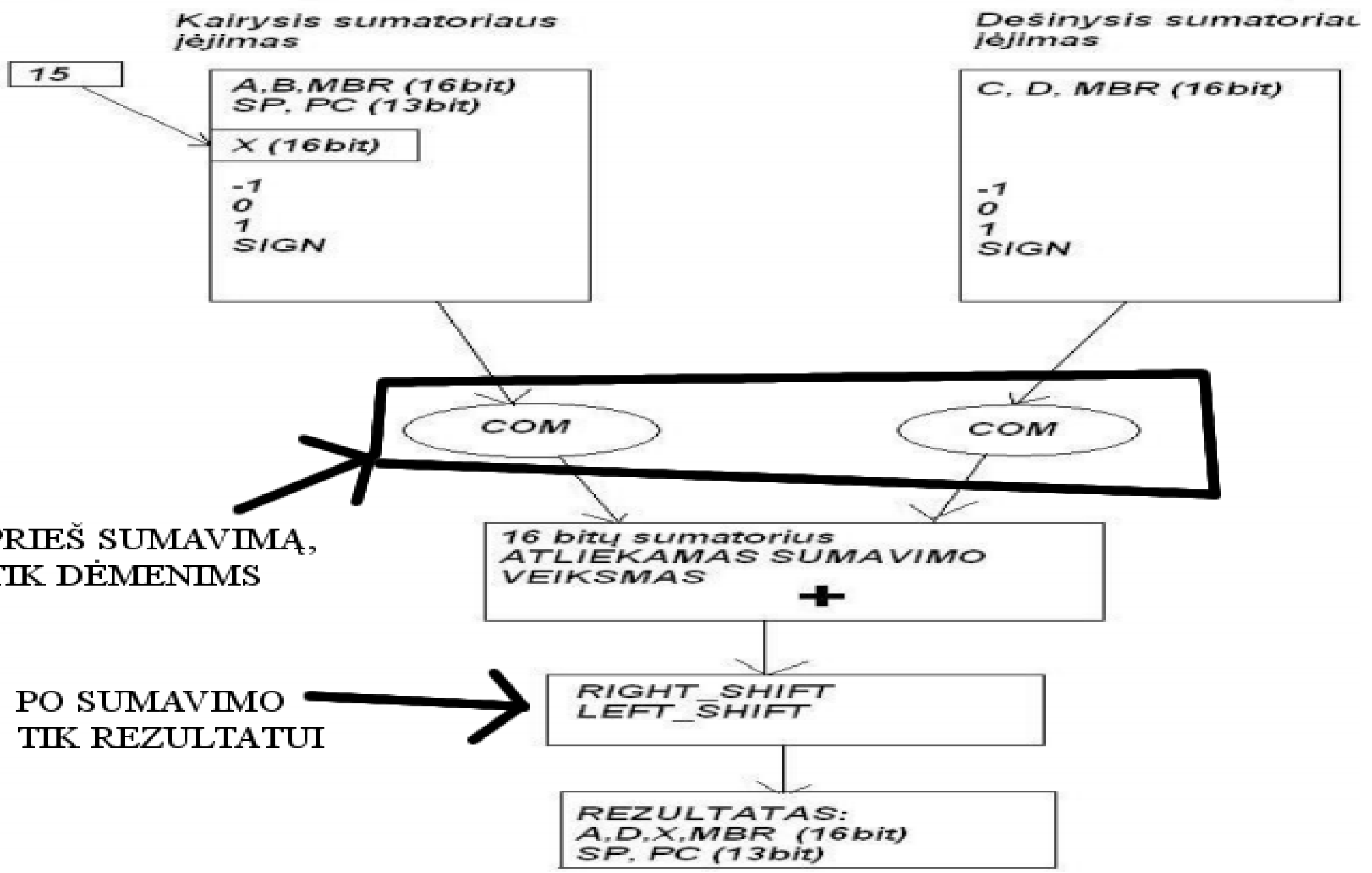
NEW SLIDE: Pusės yra LYGIAI dvi!

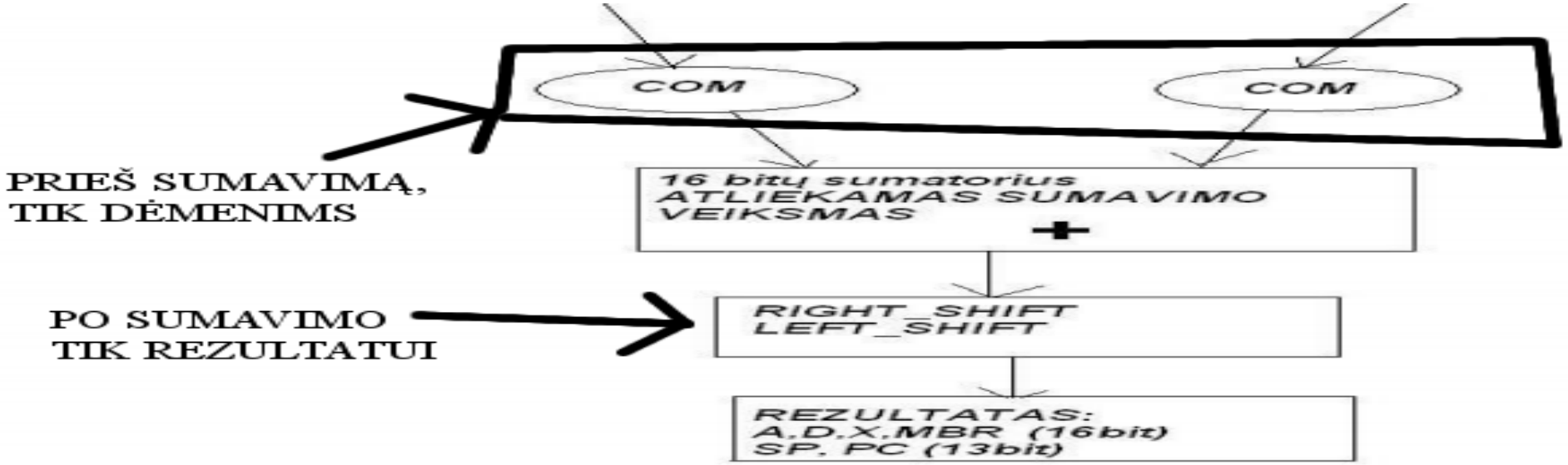
Ne minus viena, ne nulis, ne viena ir ne trys!

Jokių: A=1+1+1;



15 tiesiogiai į sumatorių nepasiųsi!


Kairysis sumatoriaus įėjimas
Dešinysis sumatoriaus įėjimas
15
A,B,MBR (16bit)
SP, PC (13bit)
X (16bit)
-1
0
1
SIGN
C, D, MBR (16bit)
-1
0
1
SIGN
COM
COM
PRIEŠ SUMAVIMĄ,
TIK DĖMENIMS
16 bitų sumatorius
ATLIEKAMAS SUMAVIMO
VEIKSMAS
+
PO SUMAVIMO
TIK REZULTATUI
RIGHT_SHIFT
LEFT_SHIFT
REZULTATAS:
A,D,X,MBR (16bit)
SP, PC (13bit)



COM funkcija gali būti taikoma tik kiekvienam iš registrų (dėmenims) atskirai, pagal poreikį, pvz:

MBR=1+COM(-1);
MBR=COM(1)+COM(-1);
MBR=COM(1)+(-1);
MBR=1+(-1);
yra teisingos komandos.

RIGHT_SHIFT ir LEFT_SHIFT gali būti taikomi tik sumavimo rezultatui:

X=RIGHT_SHIFT(COM(1)+0);
A=LEFT_SHIFT(0+COM(0));
yra teisingos komandos.

UPDATED: COM(x) – reiškia bitų invertavimą

COM(x) = - x – 1

COM(0) = -1

COM(1) = -2

COM(-1) = 0

COM(SIGN) = COM(-32768) = ***+32767*** *← buvo klaida!*

32768 = 8000h, ypatinga reikšmė, nes pakeitę ženklą (inversija ir vieneto pridėjimas) vis tiek gauname tą patį 8000h. Tai reiškia, kad 16bitų galioja ***-32768 = +32768****. Taip pat vienam baite yra su 80h. :)*

UPDATED: SHIFT'ai – reiškia, kad visi bitai pasislenka per vieną poziciją

- RIGHT_SHIFT – veikia kaip div 2
- LEFT_SHIFT – veikia kaip *2

- Išlindęs bitas pamirštamas, likusioje vietoje nuliukas.
- RIGHT_SHIFT taikant ant -1 gauname **32767**!!!

NEW SLIDE: Reminder'is

- Ir primenu dar kartą...
  - COM **tik dėmenims** (REZULTATUI NEGALIMA),

    aišku **galima**:

    A = 1+1;

    MBR = COM(A) + 1;

    juk čia COM'inamas sumavimo dėmuo, ne sumos rezultatas,

  - LEFT_SHIFT ir RIGHT_SHIFT **tik rezultatui** (vienas iš šių dviejų shift'ų, jei jo reikia). (DĖMENIMS NEGALIMA, t.y. ant kairės arba dešinės sumavimo pusės tiesiog nesideda)

    T.y. jokių:

    - RIGHT_SHIFT(LEFT_SHIFT(1+0)); ← blogai, nes galime naudoti tik vieną shift'ą vienoje mikrokomandoje arba nė vieno, jei tiesiog nereikia)
    - RIGHT_SHIFT(1)+0; ← būtų blogai, nes dėmenims shift'o taikyti negalima!

NEW SLIDE: Wild numbers are wild

STOP, and get this one into your mind:

Atliekant SHIFT'us svarbu atsižvelgt į galimus ženklo pasikeitimus:

- Kiekvienas LEFT_SHIFT potencialiai į ženklo bitą atneša naują reikšmę, nes į ženklo bito vietą ateina bitas, kuris buvo dešiniau ženklo bito prieš tai.

  1**0**10 1010 1010 1010 ***1111***

  pashiftin'us su LEFT_SHIFT pasidaro:

  0101 0101 010***1 111***0

- Kiekvienas RIGHT_SHIFT skaičių padaro teigiamu, net jei jis „atrodė“ neigiamas iki tol.

  -1 (arba 65535 jei vertinam, kaip be ženklo) tampa 32767

NEW SLIDE: Kai pradinės registrų reikšmės neduotos...

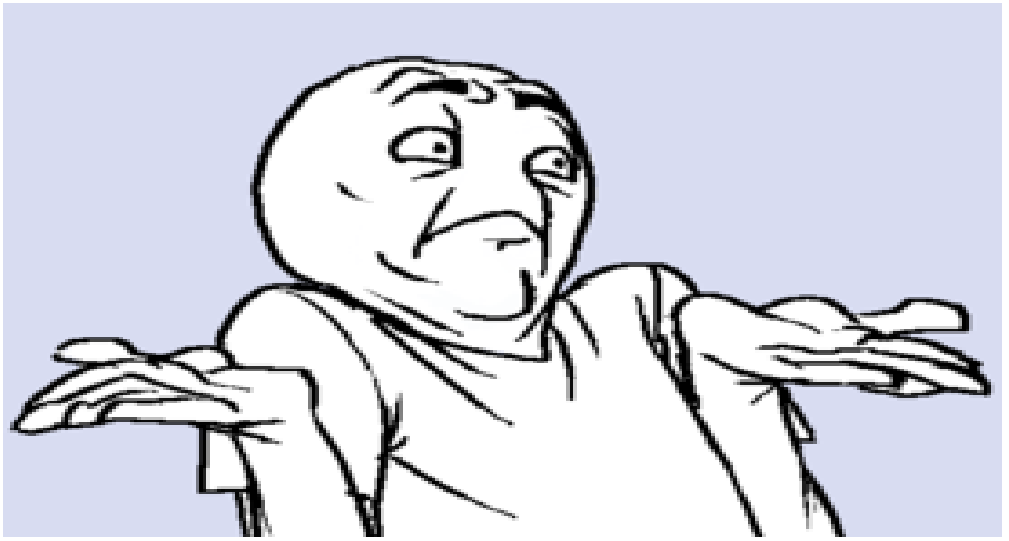
… tada spręsdami uždavinį laikome, kad nežinome kas guli, kuriame nors registre.

Todėl turime rašyti komandas taip,

kad jos veiktų vienodai nepriklausomai

nuo pradinių registrų reikšmių.

Ir tada jie manęs paklausė pradinės MBR reikšmės...



O ką aš galiu žinot...

… bet ne viskas taip blogai, yra registrų, kurių reikšmė niekada nekinta (konstantos)!

Konstantos

Išvestinės konstantos:

Veiksmas	*Šešioliktainė reikšmė*	*Dvejetainė reikšmė*	*Dešimtainė reikšmė*	
			Be ženklo	*Su ženklu*
COM(0)	FFFF	1111 1111 1111 1111	+65535	-1
COM(1)	FFFE	1111 1111 1111 1110	+65534	-2
COM(-1)	0000	0000 0000 0000 0000	0	0
COM(SIGN)	7FFF	0111 1111 1111 1111	+32767	+32767
MBR+COM(MBR)	FFFF	1111 1111 1111 1111	+65535	-1

PAPILDYTA SKAIDRĖ: Ir daugiau jokių skaičių... Viską reikia gauti iš šių...

UPDATED: O jei reikia be konstantinių...?

- Negalima naudoti: -1,0,1,SIGN,15.

- Pasinaudojame tuo, kad jei p – bet koks 16 bitų skaičius, tai

  COM(p) + p = FFFF = -1.

  Taip, čia mūsų visiškai nedomina tiksli pradinė p reikšmė.

PAPILDYTA: Respect the Kabliataškis

- Po kiekvienos operacijos dedamas ***jis***!
- Vienoje eilutėje gali būti kelios operacijos: X=15; MBR=X+1; C=MBR;
- Operacija čia vadinu veiksmą, kuris gali būti ir vienoje eilutėje, bet dar smulkiau paskaldyt neišeis:

  X=15;

  MBR=X+1;

  C=MBR;

  Ir t.t.

Rezultatą pasiųsti galima į kelias vietas iškart

Pasiunčiame skaičių 1 į A ir D registrus per vieną komandą:

A = 0 + 1; D = 0 + 1;

BET NEGALIMA: A = 1 + 0; D = 0 + 1; (nes tai reikštų sumatoriaus panaudojimą vienoje komandoje daugiau kaip vieną kartą, tai neleistina)

Netrumpinkit funkcijų pavadinimų!

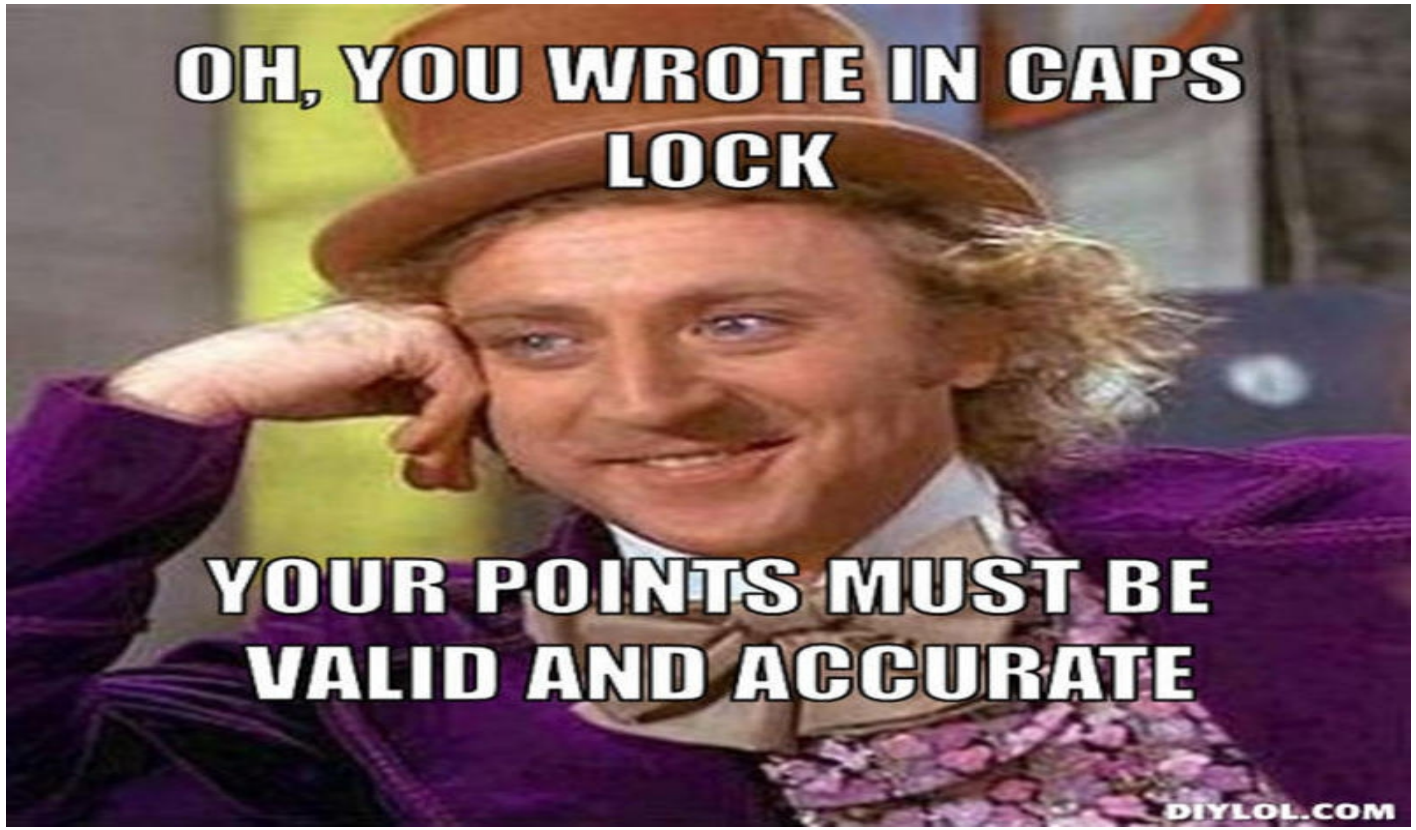
- Juokis iš savo klaidų, tai prailgins tavo gyvenimą (V. Šekspyras).
- Juokis iš mano klaidų, tai patrumpins tavo... gyvenimą. (V. Šekspyro žmona)

- Trumpink LEFT_SHIFT į SHL... Tai „*patrumpins*“ tavo pažymį.

NEW SLIDE: Rašykit didžiosiom raidėm

- Mbr = right_shIfT(a+D); gal irgi tiktų, bet...

  MBR = RIGHT_SHIFT(A+D); tiks tikrai!



Dvejetų laipsniai

- Dvejeto laipsnius bent iki $\mathbf{2^{16}}$ reikia mokėti intuityviai:

  2, 4, 8, 16, 32 … 1024 ... 32768, 65536.

  arba kitaip tariant: 2,4,8,10,20... 8000, 10000h.

Uždaviniai (1)

- Per vieną mikrokomandą į MBR pasiųskime -3.

Uždaviniai (1)

- Sprendimas:

  MBR = COM(1) + (-1);

  arba

  MBR = -1 + COM(1);

  gal reikia skliaustelių ant -1? Well...

Uždaviniai (1)

- Sprendimas:

  MBR = COM(1) + (-1);

  arba

  MBR = -1 + COM(1);

  gal reikia skliaustelių ant -1? Well...

  Don't know. Bet galima išsisukt su COM(0):

  MBR = COM(0) + COM(1);

Uždaviniai (2)

- Pasiųskime -44 į X per dvi mikrokomandas.

Uždaviniai (2)

-44 į X per dvi mikrokomandas nueina taip:

X=15; MBR=LEFT_SHIFT(COM(0) + COM(1));

X=LEFT_SHIFT(COM(X)+MBR);

Uždaviniai (3)

- Pasiųskite dešimtainę reikšmę 8192 į A ir D per dvi mikrokomandas.

Uždaviniai (3)

Gauname 8192:

A=RIGHT_SHIFT(SIGN+0);

A=RIGHT_SHIFT(A+0); D=RIGHT_SHIFT(A+0);

Savarankiškai:

- Pasiųsti -8 į A ir D per dvi mikrokomandas.
- Pasiųsti 0 į X per dvi mikrokomandas **nenaudojant konstantinių registrų!**
- Pasiųsti -48 į MBR per dvi mikrokomandas.
- Pasiųsti 7FFF į MBR per vieną mikrokomandą nenaudojant konstantinių.

- Koks rezultatas bus MBR'e įvykdžius:

  X=15; MBR=1+COM(1); C=MBR;

  MBR=RIGHT_SHIFT(COM(X)+COM(MBR));

Atsakymai

- Pasiųsti -8 į A ir D per dvi mikrokomandas.

  A=LEFT_SHIFT(COM(1)+COM(1));
  A=A+0; D=A+0;

  bet ne **0+A**!!! sumatoriaus pusės...!

Atsakymai

- Pasiųsti 0 į X per dvi mikrokomandas **nenaudojant konstantinių registrų!**

  MBR = MBR+COM(MBR);

  X = COM(MBR) + COM(MBR);

  pirmoje eilutėje galima dėmenis apkeisti

  antroj eilutėje galima uždėt shift'ą

Atsakymai

- Pasiųsti -48 į MBR per dvi mikrokomandas.

Atsakymai

- Pasiųsti 7FFF į MBR per vieną mikrokomandą nenaudojant konstantinių.

  MBR = RIGHT_SHIFT(MBR+COM(MBR));

  arba

  MBR= RIGHT_SHIFT(COM(MBR)+MBR);

Atsakymai

- Koks rezultatas bus MBR'e įvykdžius:

  X=15; MBR=1+COM(1); C=MBR;

  MBR=RIGHT_SHIFT(COM(X)+COM(MBR));

  Sprendimas:

  X=15; MBR=1-2=-1; (*C=MBR* ← *nieko nekeičia*)
- MBR = RIGHT_SHIFT(FFF0 + 0000); t.y. RIGHT_SHIFT ant FFF0, kas yra... ***7FF8***