**KAUNO TECHNOLOGIJOS UNIVERSITETAS**

**INFORMATIKOS FAKULTETAS**

**Modulio P175B125 „Kompiuterių architektūra“**

Laboratorinio darbo aprašas (ataskaita)

**Pirmas laboratorinis darbas (tema 1.3. Procesorius. Valdymo ir operacinis įtaisai)**

|  |  |
| --- | --- |
|  | **Dėstytojas**  MACIULEVIČIUS Stasys  doc. ČINČIKAS Gedeiminas |
|  |
| **Studentas**  Kuzmickas Lukas |

**KAUNAS, 2022**

Turinys

[1. Užduoties analizė 3](#_Toc115612900)

[2. Algoritmo medis 4](#_Toc115612901)

[3. simuliacijos rezultatai 6](#_Toc115612902)

[4. Rezultatų analizė 9](#_Toc115612903)

# Užduoties analizė



1 pav. Individualios užduoties variantas

Spręsime tokią (170nr.) užduotį: reikia paskaičiuoti išraiškos (1 pav.)

F = (a-b) /(c×c), rezultatą, kai a > 0, b < 0, c > 0. Šiame darbe nerealizuojame dalybos, todėl gauname tik (a-b) ir (c×c) rezultatus. Kadangi b < 0 (visada neigiamas), tai darome išvadą,

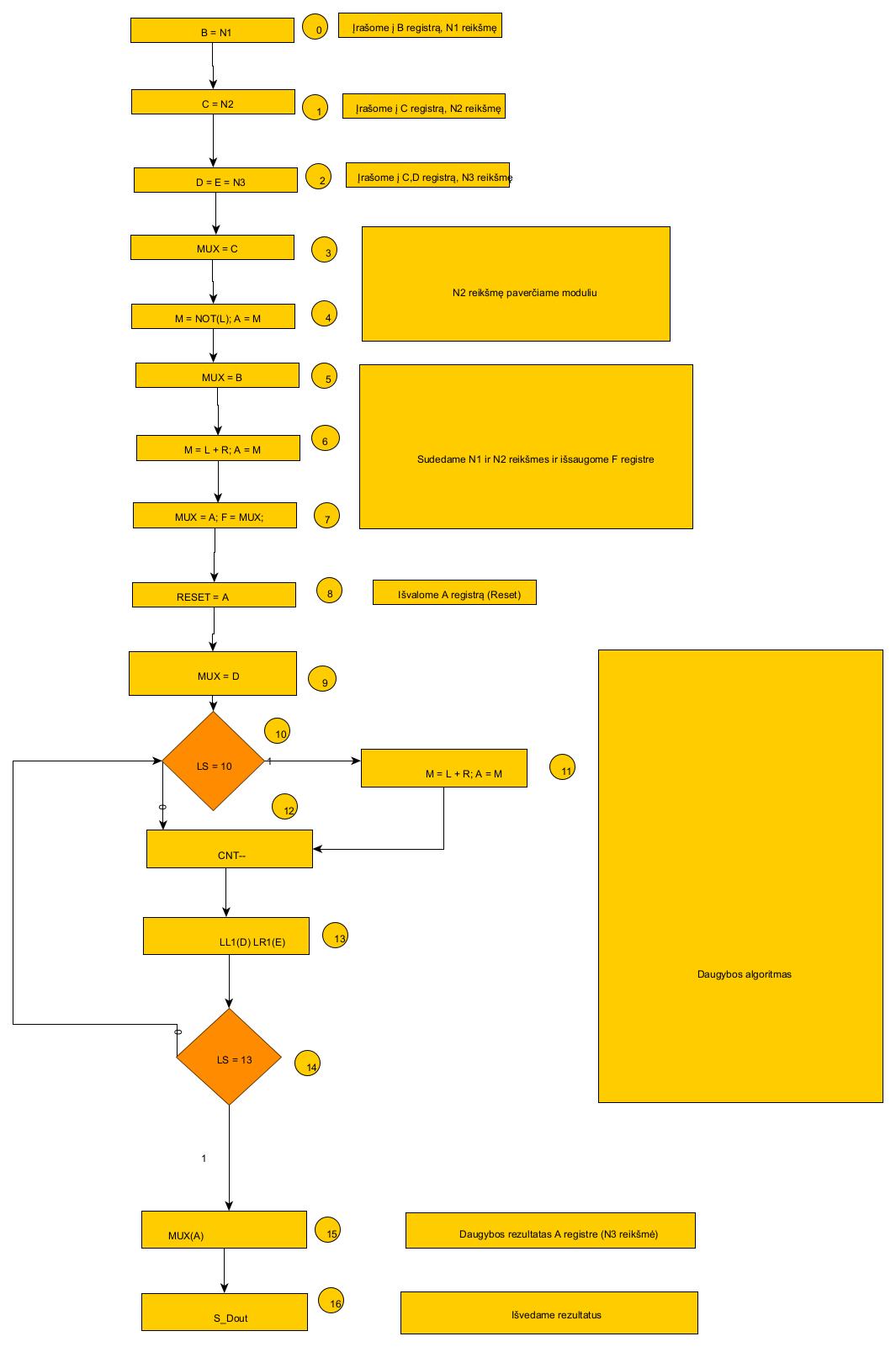
kad (a-(-b)) visada lygu a+b, kadangi minusas kart minusas lygu pliusui. Taipogi, c>0, tai sandauga visada bus teigiamas skaičius. Mūsų adresacija – natūrali, o pateiktas kodas atvirkštinis.

Šioje užduotyje realizavome tik sudėties ir daugybos operacijas – atsižvelgdami į ženklą ir kodą.

# Algoritmo medis

Šiam labaratoriniui darbui susidarėme algoritmo medį (2 paveikslėlis), pagal kurį užpildėme mūsų norimas mikrooperacijas. Iš pradžių į registrus B,C,D,E per DIN įrašome mūsų norimas reikšmes atvirkštiniu kodu (0,1,2 mikrooperacijos). Kadangi mūsų N2 reikšmė C registre yra neigiama, o pagal sąlyga:

N1-(-N2) == N1+N2, ji visada bus teigiama, todėl ją privalome invertuotį ir gauname šio skaičiaus modulį, t.y. teigiamą reikšmę. (3 mikrooperacija – nuskaitome C registro reikšmę į MUX, 4 mikrooperacija – naudodami ALU invertuojame šį skaičių ir gauname teigiamą skaičių, kuris išsaugojamas A registre – ALU R reikšmė). Po šito veiksmo nuskaitome B registro skaičių į MUX ir paduodame į ALU, kur vyksta sudėties operacija (L+R), gautą rezultatą išsaugojame į F registrą (5,6,7 mikrooperacijos). Paskui, pasiruošiame daugybai, išvalome A registrą, nes jis bus naudojamas, kaip akumaliatorinis registras (8 mikrooperacija). Vykdome daugybos algoritmą, aprašydami loginiais flagais. Nuskaitome D registro duomenis į MUX. Jeigu E registro mažiausias bitas nėra lygus 1, atliekame loginio postūmio operacija LL1(D) ir LR1(E) registruose. Kol mūsų skaitiklis nėra tuščias (LS = 13), šios operacijos kartojamos – pabaigoje sudedami registrai ir daugybos rezultatas (N3×N3) išvedamas į A registrą. (14,15 mikrooperacijos)



2 pav. Algoritmo medis

# simuliacijos rezultatai

Naudojamos direktyvos:

restart -f

force -freeze sim:/top/RST 1 0, 0 {5 ps}

force -freeze sim:/top/CLK 0 0, 1 {10 ps} -r 20

force -freeze sim:/top/Din 0000000000000101 0

force -freeze sim:/top/Din 1111111111110011 30

force -freeze sim:/top/Din 0000000000001000 50

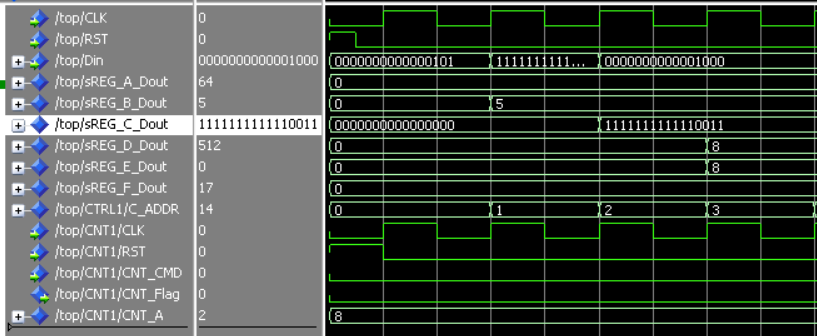
run 3000

N1 = 5;

N2 = -12;

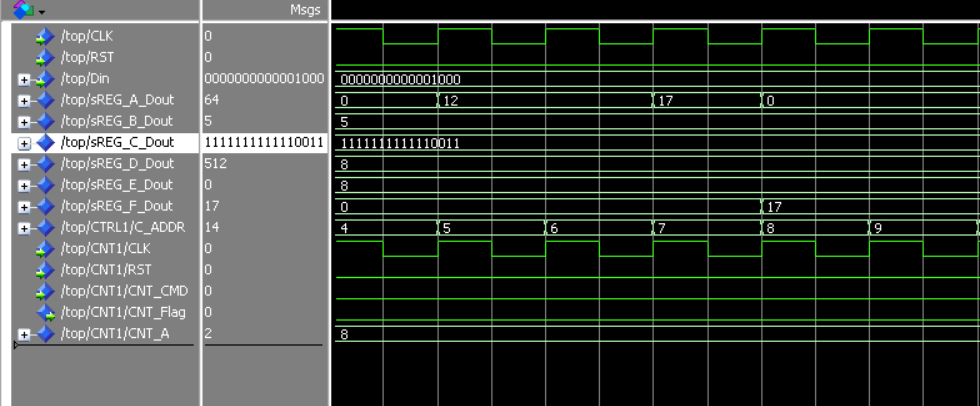
N3 = 8;

Iš 3 paveikslėlio matome duomenų įvedimą, tam tikro C\_ADDR reikšmės metu. N1 reikšmė įvedama (B registras – B\_Dout), kai C\_ADDR = 0, N2 reikšmė (C registras – C\_Dout), kai C\_ADDR = 1 ir N3 reikšmė (D,E registrai – D\_Dout, E\_Dout) įvedama, kai C\_ADDR = 3.



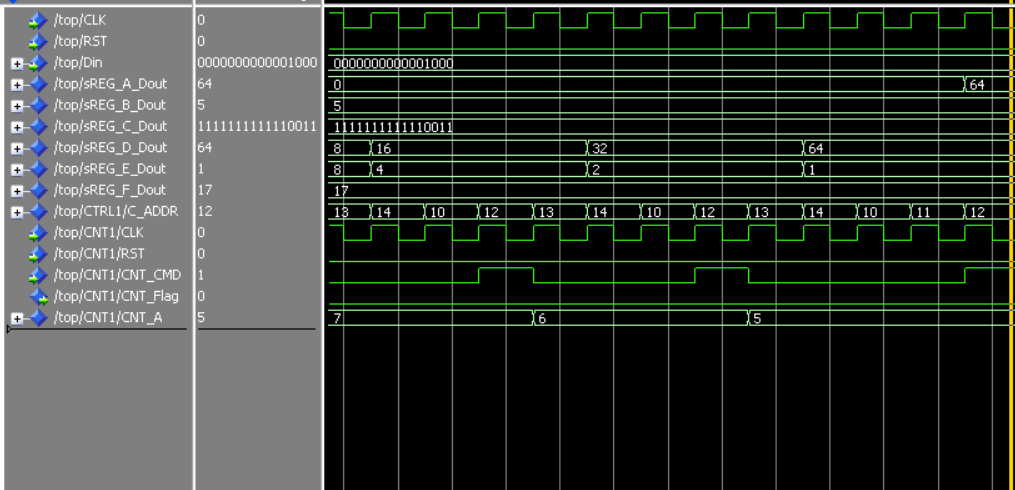
3 pav. Duomenų įvedimas

Iš 4 paveikslėlio matome sudėties operacija: N1-(-N2). Kai C\_ADDR = 4, invertuojame C registro reikšmę ir gauname teigiamą skaičių modulį (iš atvirkštinio kodo), kuris įrašomas į A\_Dout registrą. Kai C\_ADDR = 6, atliekame sudėties operacija (5+12=17), šį rezultatą įrašomę į F\_Dout registrą (kai C\_ADDR = 7). Pabaigoje C\_ADDR = 8, išvalome A registrą (A\_Dout) paruošdami daugybos algoritmui. (5-(-12)) = 17. Rezultatas teisingas ir grąžinamas atvirkštiniu kodu.



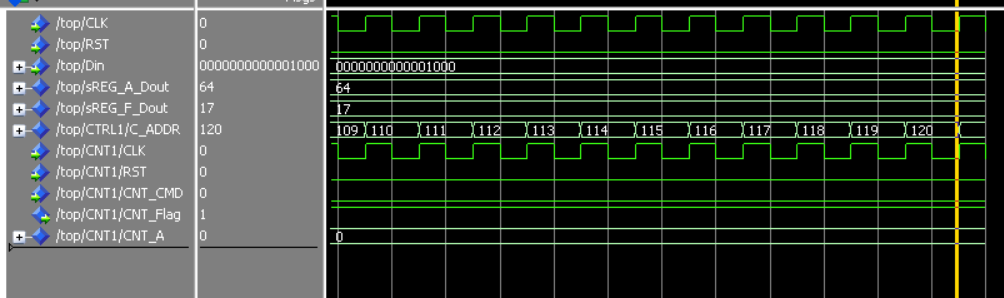
4 pav. Sudėties operacija

Iš 5 paveikslėlio matome daugybos algoritmą, CNT\_A skaitliukas mažėja. Matome, kol nepatenkinama LS=10 (E registro L=1), tol adresacija keičiasi tarp C\_ADDR = 10 => 12 => 13 ir kol skaitliukas nėra tuščias LS=13 peršoka į C\_ADDR = 14 => 10. Viskas vykdoma, kol tenkinama LS = 10 sąlyga ir rezultatas sudedamas ir įrašomas į A registrą (A\_Dout). N3 = 8. N3\*N3 = 64. Rezultatas teisingas ir grąžinamas atvirkštiniu kodu.



5 pav. Daugybos algoritmas

Galutiniai rezultatai išvedami A ir F registruose (6 paveikslėlis). A registre saugomas N3\*N3 rezultatas. F registre N1-(-N2) rezultatas. Visi rezultatai juose saugomi atvirkštiniu kodu.



6 pav. A ir F registro reikšmės

# Rezultatų analizė

Šioje individualioje užduotyje sėkmingai realizavome sudėties ir daugybos operacijas. Atliekame sudėties operacija N1-(-N2) įvertinant ženklą ir atvirkštinį kodą – atliekame reikalingas konversijas ir gražiname rezultatą formatu, kokiu jis buvo gautas – t.y. atvirkštiniu kodu.

Realizavome ir daugybą, įvertinant, kad ženklas visada teigiamas – rezultatas grąžinamas tokiu formatu, kokiu ir buvo gauti duomenys – t.y. atvirkštiniu kodu.

Išvados:

* Realizuotos sudėties ir daugybos operacijos, įvertinant ženklą ir kodą.
* Gaunami korektiški atsakymai, pagal pateiktus duomenis.
* Rezultatai grąžinami atvirkštiniu kodu.
* Suprasti esminiai procesoriaus operacijų mikroprogramavimo pagrindai.