**Mikroprogramos sudarymo pavyzdys**

Rekomenduojama tokia užduoties sprendimo tvarka:

Koregavimas

Užduoties analizė

Algoritmo sudarymas

Struktūros analizė

Mikroprogramos kodavimas analizė

Mikroprogramos testavimas analizė

Mikroprogramos sudarymas analizė

1 pav. Užduoties sprendimo tvarka

Plačiau paaiškinsime kiekvieną žingsnį, iliustruodami sprendžiamu pavyzdžiu.

**Užduoties analizė**

Spręsime tokią užduotį: paskaičiuoti išraiškos

**F = (a×b)/(a+b)**

rezultatą, kai **b**<0, **a**>|**b**|, neigiami skaičiai pateikiami papildomuoju kodu (rezultatas turi būti pateiktas taip pat papildomuoju kodu).

Kadangi **a**>|**b**|, suma bus teigiama, sandauga **a×b** – neigiama, rezultatas - taip pat neigiamas.

**Algoritmo sudarymas**

Algoritmą būtų galima apibūdinti kaip **tikslių nurodymų seką tam, kas turės atlikti konkrečią užduotį**. Algoritmas turi patenkinti šias sąlygas:

* jis turi atlikti darbą;
* jis turi būti aiškus ir nedviprasmiškas;
* jis turi apibrėžti žingsnių seką, reikalingą darbui atlikti, t. y. jis turi nurodyti žingsnių atlikimo tvarką;

Informatikoje dažnai dar reikalaujama, kad algoritmas būtų baigtinis dviem prasmėm:

* atliekamų žingsnių skaičius turi būti baigtinis, t. y. algoritmas turi tikrai baigti darbą;
* kiekvienam žingsniui atlikti turi pakakti baigtinio laiko ir baigtinių resursų, t. y. kiekvienas žingsnis turi būti toks, kad jį būtų galima atlikti.

Pastarieji du reikalavimai garantuoja, kad algoritmas bus baigtas baigtiniu laiku ir su baigtiniais resursais. Algoritmai, tenkinantys tik pirmąsias tris sąlygas, vadinami daliniais algoritmais, o tenkinantys visas penkias sąlygas – pilnais algoritmais.

Užduoties sprendimo (grubus) algoritmas pateiktas 2 paveiksle. Paaiškinsime jo žingsnius:

1. Įvedame pradinius duomenis papildomuoju kodu.
2. Kadangi dauginsime skaičių modulius, daugybai paruošiame operandus, į **c** įrašydami –**b**. Tam teks atlikti veiksmus **c**=not(**b**)+1.
3. Pirmiausia paskaičiuosime vardiklio **a+b** reikšmę. Ją perstumsime į žodžio pradžią, nes to reikalauja dalybos algoritmas.
4. Vykdant dalybą, vardiklis atimamas arba pridedamas, todėl reikia turėti ir neigiamą vardiklio reikšmę.
5. Paskaičiuosime skaitiklio **a**×**c** reikšmę.
6. Atliksime dalybą, naudodami skaitiklio ir vardiklio modulius.
7. Suformuosime galutinį rezultatą. Kadangi sandauga yra neigiama, o suma bus teigiama, rezultatas bus neigiamas skaičius, todėl reikės jį išvesti papildomuoju kodu.

Įvedame a, b

c = -b

Paskaičiuojame v=a+b ir perstumiame į kairę

Suformuojame neigiamą v reikšmę

Paskaičiuojame s=a×c

Paskaičiuojame F=s/v

Pakeičiame F ženklą, kodą ir išvedame

2 pav. Užduoties sprendimo (grubus) algoritmas

Toliau pateiksime modulių daugybos ir dalybos algoritmus.

***Daugybos algoritmas***

Paprastumo dėlei dauginsime skaičių modulius, tad korekcijų nereikės. Daugybos nuo žemiausiųjų skilčių stumiant dauginamąjį algoritmas pateiktas 3 paveiksle. Paaiškinsime jo žingsnius:

1. Nustatome pradines sandaugų sumos **s** ir skaitiklio **sk** reikšmes.
2. Tikriname žemiausiąją **c** skiltį (**cl**).
3. Jei ji lygi 1, prie **s** pridedame **a**.
4. Daugiklį **c** pastumiame į dešinę, dauginamąjį **a** – į kairę. Sumažiname skaitiklio **sk** turinį.
5. Kol skaitiklis **sk** nelygus 0, grįžtame į 2 žingsnį.

s = 0, sk = 8

cl = 1?

s = s + a

ll1(a), lr1(c), sk = sk - 1

sk = 0?

Taip

Taip

Ne

Ne

r = 0, sk = 8

L = L + (- v), ll1(r)

L < 0?

Ne

Taip

sk <=0? 0?

Taip

Ne

L = L + v

r = r + 1

ll1(L), sk = sk - 1

3 pav. Modulių daugybos algoritmas 4 pav. Modulių dalybos algoritmas

***Dalybos algoritmas***

Paprastumo dėlei dalinsime skaičių modulius. Dalybos atstatant liekaną algoritmas pateiktas 4 paveiksle. Paaiškinsime jo žingsnius:

1. Nustatome pradines rezultato **r** ir skaitiklio **sk** reikšmes. Pradinė liekana **L** bus dalijamasis – skaitiklis **s**.
2. Iš liekanos **L** atimame daliklį (vardiklį) **v** ir rezultatą **r** pastumiame į kairę, tuo pačiu suformuodami eilinę dalmens skiltį, lygią nuliui.
3. Tikriname liekanos **L** ženklą.
4. Jei liekana neigiama, ją atstatome, prie **L** pridėdami **v**.
5. Jei liekana teigiama, prie **r** pridedame 1, tuo pakeisdami eilinę dalmens skiltį vienetu.
6. Liekaną **L** pastumiame į kairę. Sumažiname skaitiklio **sk** turinį.
7. Kol skaitiklis **sk** nelygus 0, grįžtame į 2 žingsnį.

Šį dalybos atstatant liekaną algoritmą modifikuosime, atsižvelgdami į eilinės dalmens skilties formavimo ypatumus struktūroje. Modifikuotas algoritmas pateiktas 5 paveiksle.

Pastumti v ir (-v) į kairę 7 kartus, r – 8 kartus

L = L + (- v), ll1(r)

L < 0?

Ne

Taip

sk <=0? 0?

Taip

Ne

LL1(r), L = L + v

CL1(r)

LL1(L), sk = sk - 1

Suformuoti r reikšmę lygią FFFF

sk = 8

5 pav. Modifikuotas modulių dalybos algoritmas

Paaiškinsime jo žingsnius:

1. Reikiamai pradinei rezultato **r** reikšmei formuoti **r** priskiriame reikšmę FFFF.
2. Pastumiame vardiklio teigiamą (**v**) ir neigiamą (-**v**) reikšmes į kairę 7 kartus, o **r** – 8 kartus.
3. Nustatome skaitiklio **sk** reikšmę. Pradinė liekana **L** bus dalijamasis – skaitiklis **s**.
4. Iš liekanos **L** atimame daliklį (vardiklį) **v** ir rezultatą **r** pastumiame į kairę.
5. Tikriname liekanos **L** ženklą.
6. Jei liekana neigiama, ją atstatome, prie **L** pridėdami **v**, o **r** pastumiame į kairę **logiškai**.
7. Jei liekana teigiama, **r** pastumiame į kairę **cikliškai**.
8. Liekaną **L** pastumiame į kairę. Sumažiname skaitiklio **sk** turinį.
9. Kol skaitiklis **sk** nelygus 0, grįžtame į 2 žingsnį.

**Struktūros analizė**

Kintamiesiems saugoti naudosime tokius registrus:

* **a** įvesime į registrą B;
* **b** įvesime į registrą E;
* |**b**| laikysime registre C;
* paskaičiavę **a**+**b**, įrašysime į registrą E;
* **-**(**a**+**b**) įrašysime į registrą D;
* daugybos metu dalinių sandaugų sumą kaupsime registre A;
* dalybos metu liekaną laikysime registre A;
* dalmenį **r** kaupsime registre F.

**Mikroprogramos sudarymas**

VHDL aprašytos struktūros ypatumai reikalauja kai kuriuos atrodytų paprastus žingsnius realizuoti ne viena, o keliomis mikrokomandomis. Pavyzdžiui, jei reikia susumuoti dviejų registrų B ir C turinius, įrašant sumą į registrą D, reikės tokių mikrokomandų:

|  |  |  |
| --- | --- | --- |
|  | Prasmė | Mikrooperacijos |
| MK1 | A = 0 | Reset. RegA |
| MK2 | A = A + B | Mux. RegB, ALU.L+R, RegA.ALU\_M |
| MK3 | A = A + C | Mux. RegC, ALU.L+R, RegA.ALU\_M |
| MK4 | D = A | Mux. RegA, RegD.Mux |

Čia kiekviena mikrooperacija sąlyginai užrašyta nurodant komponentą ir to komponento mikrooperaciją, atskiriant tašku.

Mikroprogramai sutrumpinti galima panaudoti jos pirmosios ir paskutinės mikrokomandų suliejimą su kitomis. Pavyzdžiui, jei prieš MK1 esančioje MK registras A nereikalingas, A nustatymą į 0 galima įdėti į ją.