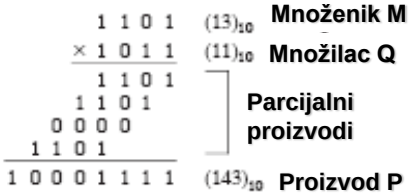
Množenje neoznačenih brojeva

- Množenje je mnogo kompleksnija operacija od sabiranja/oduzimanja
- Primer "ručnog" množenja



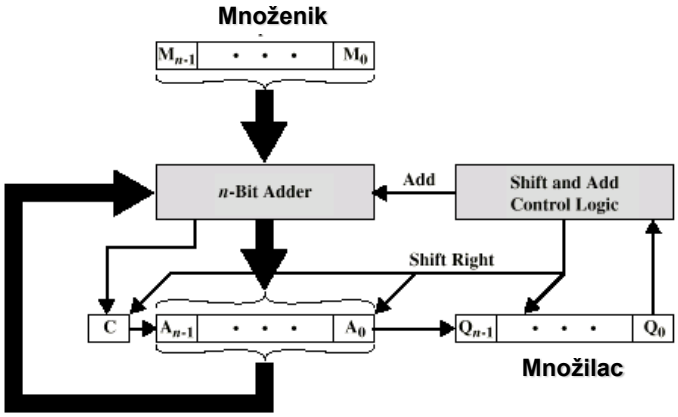
- Množemo zaključiti sledeće
1. U toku množenja generišu se parcijalni proizvodi za svaku cifru množioca.
   - Parcijalni proizvodi se na kraju sumiraju da bi se dobio rezultat
2. Parcijalni proizvodi se lako određuju
   - Ako je odgovarajuća cifra množioca 0, parcijalni proizvod je 0
   - Ako je odgovarajuća cifra množioca 1, parcijalni proizvod je jednak množeniku
3. Svaki parcijalni proizvod se mora pomeriti ulevo za jednu poziciju u odnosu na prethodni parc. proizvod
4. Množenje dva n-to bitna broja generiše rezultat koji može imati do 2n bitova

Množenje neoznačenih brojeva –nast.

- U odnosu na postupak "ručnog" množenja uz pomoć papira i olovke, množenje na računaru se razlikuje po tome što se
  - sumiranje parcijalnih proizvoda obavlja odmah, a ne na kraju
  - Množenje jedinicom zahteva pomeranje ulevo i sabiranje
  - Množenje nulom zahteva samo pomeranje ulevo

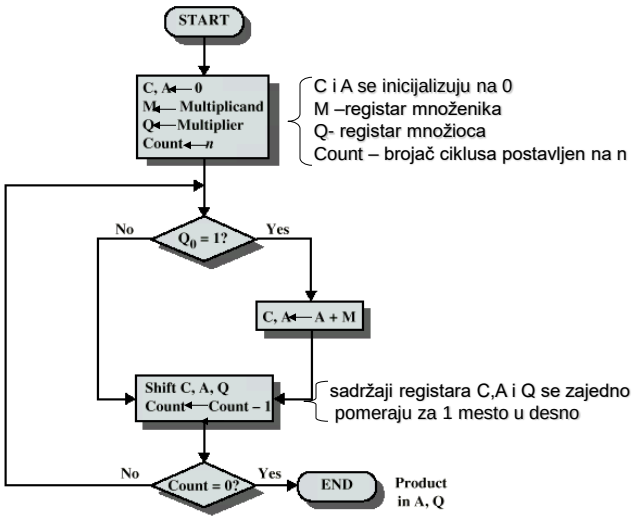
Blok dijagram množača neoznačenih brojeva

- Tri registra
  - Registar množenika M
  - Registar množioca Q
  - Registar A koji je inicijalizovan na 0
  - Jednobitni registar C za pamćenje bita prenosa ako se pojavi u toku sabiranja parcijalnih proizvoda
  - Rezultat se dobija zajedno u registru A i Q



(a) Block Diagram

Dijagram toka množenja neoznačenih brojeva



Primer: sadržaji registara tokom množenja

Množi se 1011x1101 (11x13=143)

C	A	Q	M		
0	0000	1101	1011	Initial Values	
0	1011	1101	1011	Add	First Cycle
0	0101	1110	1011	Shift	
0	0010	1111	1011	Shift	Second Cycle
0	1101	1111	1011	Add	Third Cycle
0	0110	1111	1011	Shift	
1	0001	1111	1011	Add	Fourth Cycle
0	1000	1111	1011	Shift	

rezultat

Šta je sa množenjem označenih brojeva?

- Ako se koristi predstavljanje u komplementnom obliku za negativne brojeve, neće se dobiti korektan rezultat
- Rešenje:
  - koristiti prosto označavanje brojeva
    - bitovi najveće težine koriste se za predstavljanje znaka
  - znak proizvoda zavisi od znakova množenika i množioca:
    - ako su istog znaka, rezultat je pozitivan
    - ako su različitog znaka rezultat je negativan

Kako ubrzati množenje?

- Množenje dva n-to bitna broja zahteva da se obavi n-1 operacija pomeranja i n-1 sabiranje
- Operacija pomeranja se lako implementira i brže se izvršava od sabiranja
  - Broj sabiranja se može smanjiti ako se ima u vidu da se broj koji sadrži n uzastopnih jedinica može predstaviti kao razlika dva broja

$$(\ldots 0\overbrace{1\ldots 1}^{n}0\ldots)_2 \equiv (\ldots 1\overbrace{0\ldots 0}^{n}0\ldots)_2 - (\ldots 0\overbrace{0\ldots 1}^{n}0\ldots)_2.$$

  - Npr. Ako je množilac 01111 = 10000 -0001, umesto 3 sabiranja, množenje se može obaviti samo pomeranjem i jednim oduzimanjem
    - Npr. dekadno 75x999 =75x(1000 -1)= 75000-75=74925

Booth-ov algoritam

- Ovo zapaženje iskoristio je engleski matematičar Andrew Donald Booth i 1951 predložio algoritam koji se može iskoristiti za blokove jedinica bilo koje dužine, pa i dužine 1.
  - Booth-ov algoritam ne mora uvek dovesti do smanjenja broja operacija sabiranja
    - Npr. Ako bi množilac bio oblika 101010, primena algoritma bi udvostručila broj sabiranja
- Pokazalo se da se Booth-ov algoritam može iskoristiti za množenje označenih brojeva predstavljenih u dvoičnom komplementu

Booth-ov algoritam

- Booth-ov algoritam ispituje susedne parove bitova n-bitnog množioca Q predstavljenog u dvoičnom komplementu, uključujući i implicitni bit $Q_{-1}=0$

Q_i	Q_{i-1}	akcija
0	0	ništa
0	1	Saberi množenik pomeren za i mesta ulevo sa parcijalnim proizvodom (detektuje se kraj bloka jedinica)
1	0	Oduzmi množenik pomeren za i mesta ulevo sa parcijalnim proizvodom (detektuje se početak bloka jedinica)
1	1	ništa

Booth-ov algoritam primer

množenik M= 0111 (7), množilac Q =0011 (3)

A	Q	Q_{-1}	M		
0000	0011	0	0111	Initial Values	
1001	0011	0	0111	A ← A - M	First Cycle
1100	1001	1	0111	Shift	
1110	0100	1	0111	Shift	Second Cycle
0101	0100	1	0111	A ← A + M	Third Cycle
0010	1010	0	0111	Shift	
0001	0101	0	0111	Shift	Fourth Cycle

oduzimanje se svodi na sabiranje komplementa

Rezultat
7x3=21

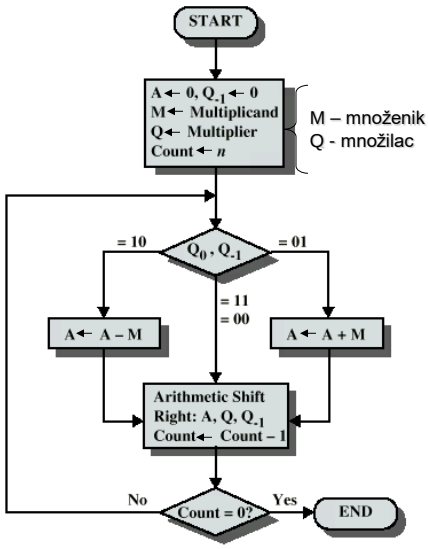
Booth-ov algoritam primer 2

množenik M= 0111 (7), množilac Q =-3 = 1101

A	Q	Q-1	M	
0000	1101	0	0111	
1001	1101	0	0111	A:=A-M (oduzimanje se svodi na sabiranje komplemenata)
1100	1110	1	0111	shift right (aritmetičko pomeranje)
0011	1110	1	0111	A:=A+M
0001	1111	0	0111	shift right
1010	1111	0	0111	A:=A-M (oduzimanje se svodi na sabiranje komplemenata)
1101	0111	1	0111	shift right (aritmetičko pomeranje)
1101	0111	1	0111	ništa se ne radi
1110	1011	1	0111	shift right (aritmetičko pomeranje)

1110 1011 = -21

Booth-ov algoritam



Hardverska struktura množioca

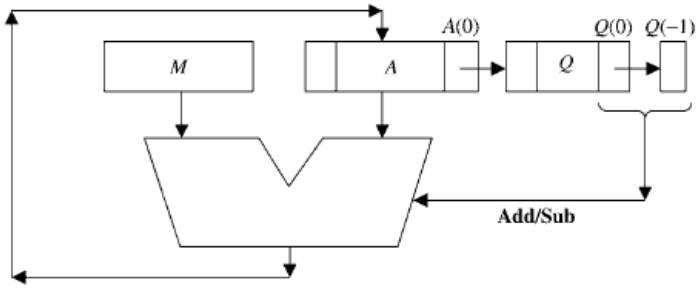
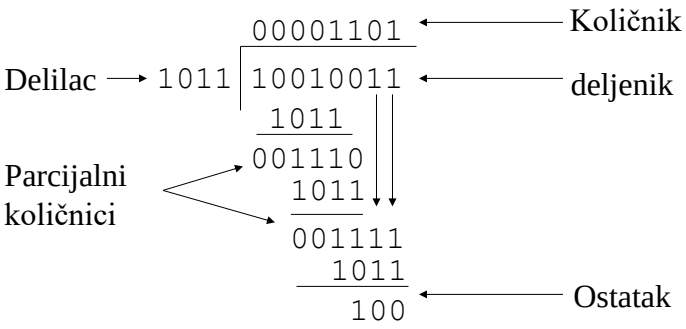


Figure 4.9 Hardware structure implementing Booth's algorithm

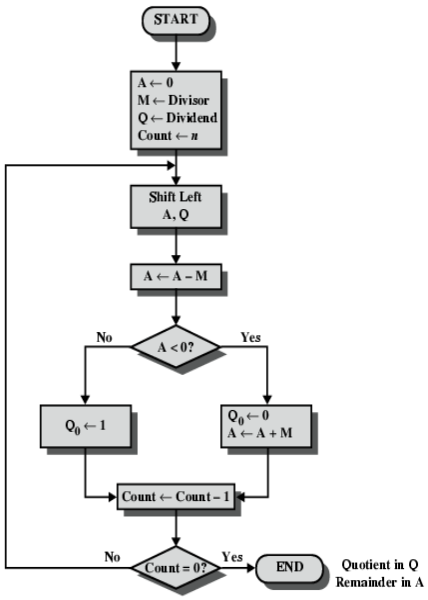
Celobrojno deljenje

- Malo složenije od celobrojnog množenja, ali se bazira na istim opštim pricipima
  - Svodi se na operacije pomeranja i sabiranja ili oduzimanja
  - Opet analogija sa "ručnim" deljenjem
    1. Prvo se bitovi deljenika se pretražuju s leva u desno dok se ne pronađe niz bitova koji je veći ili jednak deliocu
       - Kaže se da delitelj može da deli broj
    2. Dok se ne pronađe ovaj niz 0 se upisuju u količnik s leva u desno
    3. Kada se pronađe niz koji zadovoljava uslov 1. upisuje se 1 u količnik, i delitelj se oduzima od parcijalnog deljenika
       - Rezultat dobijen oduzimanjem predstavlja parcijalni ostatak
    4. Od ove tačke pa nadalje, proces se ciklično ponavlja
       - U svakom ciklusu sledeći bitovi deljenika se nadovezuju na parcijalni ostatak dok se ne dobije broj koji je veći ili jednak deliocu
       - Zatim se delilac oduzima, itd.
       - Proces se okončava kada se iscrpu svi bitovi deljenika

Celobrojno deljenje -primer



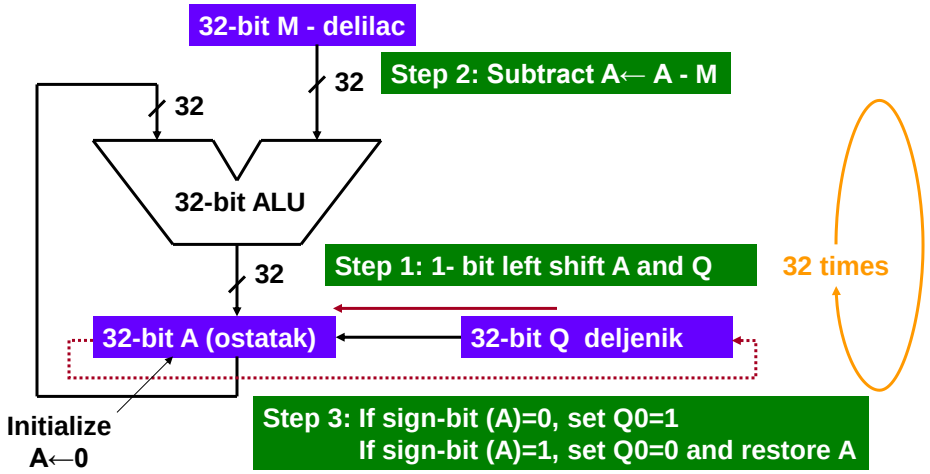
Dijagram toka algoritma za celobrojno deljenje



- Delilac je u registru M,
- Deljenik je u registru Q
- Count dužina operanada

* Na kraju procesa količnik je u registru Q, a ostatak u registru A;
  - A je inicijalno 0
* U svakom koraku sadržaji registra A i Q se zajedno pomeraju ulevo za jedno mesto
* M se oduzima od A da bi se proverilo da li M deli parcijalni ostatak
  - ako je A <0, regeneriše se parcijalni ostatak

Implementacija deljenja



U registru Q se dobija količnik, a u registru A ostatak.