6. OPERACIJE NAD PODACIMA

6.1. Klasifikacija operacija

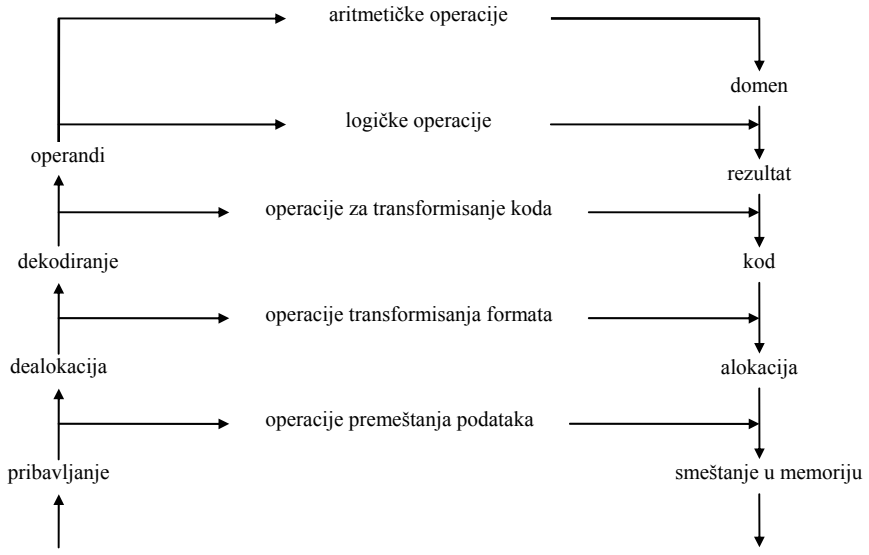
Već smo ranije uočili da postoji razlika između elemenata podataka kao apstraktnog koncepta i prezentacije elemenata podataka u bitovima. Na sličan način, apstraktna operacija, koja ukazuje na efekat operacije nad apstraktnim elementima podataka, razlikuje se od efekta koji mašinska operacija ima nad prezentacijom bit oblika. Ove apstraktne operacije imaju jednu dobru osobinu koja se ogleda u tome što su veoma bliske konceptu kako čovek razmišlja o tim operacijama. One takođe obezbeđuju da se operacije mogu posmatrati izdvojeno od prezentacije podataka, što omogućava ortogonalni tretman oba predmeta.

Apstraktna definicija mašinske operacije mora da sadrži sledeće delove:

- apstrakciju željenih operanada iz dostupnih bit oblika,
- opis same apstraktne operacije
- preslikavanje apstraktnih rezultata na dostupne bit oblike.

Svaki od ovih delova se može dalje deliti na korake, sl. 6.1, koji se mogu objasniti na sledeći način:

- Prezentacija podataka, koju čine bit oblici, mora da se pribavi iz memorije.
- Pribavljeni bit oblici se dele na posebna polja u saglasnosti sa odgovarajućim pravilima (na primer, FP brojevi se dele na eksponent i frakciju).
- Da bi se odredila vrednost operanada, polja se moraju dekodirati.
- Slede operacije koje se obavljaju nad jednim ili većim brojem operanada.
- Svaka operacija generiše jedan ili veći broj rezultata. Kada su operacije aritmetičkog tipa, moguće je da rezultat bude van opsega korišćene prezentacije brojeva.
- Rezultat se mora kodirati da zadovolji specifikacije željene prezentacije.
- Kodirana polja se moraju upakovati u bit oblik u saglasnosti sa pravilima koja važe za prezentaciju formata podataka.
- Kada su bit oblici alocirani, pridružuje im se informacija o memorijskoj lokaciji (ili registru) i smeštaju se u memoriju.



Sl. 6.1. Tipovi operacija.

Saglasno slici 6.1 tipovi operacija se mogu klasifikovati na sledeći način:

- **Operacije premeštanja podataka** (Data Move Operations). Alokacija i kodiranje nemaju uticaj na ove operacije; samo se menja međusobni odnos ime-podatak. Na primer kod MC68020 postoji instrukcija

  MOVE Src,Dst

  čiji je efekat da sadržaj memorijske lokacije određene od strane odredišta bude jednak sadržaju memorijske lokacije određene od strane izvorišta (Dst:=(Src)).
- **Operacije koje obuhvataju transformaciju formata** (Format Transformation Operations). Korišćenje ovih operacija nema uticaj na kodiranje podataka; samo se menja lokacija bitova. Na primer, kod MC68020 postoji instrukcija

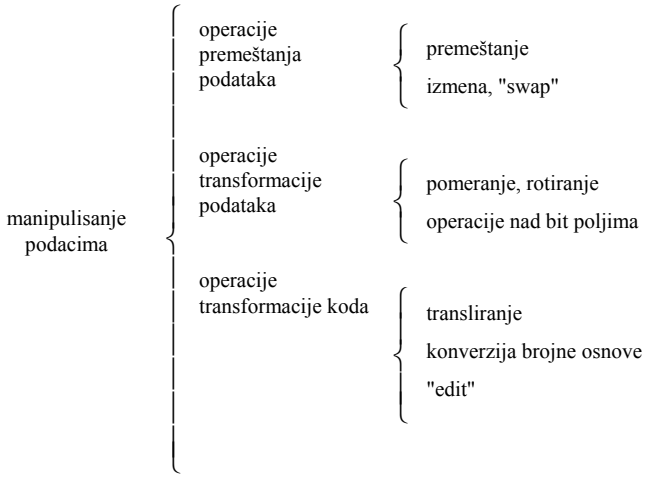
  LSL Cnt,Dst

  Efekat ove instrukcije se ogleda u tome što pomera odredišni operand za Cnt bit pozicija, dok se nula bitovi ubacuju sa desne strane.
- **Transformacije koda** (Code Transformations). To su operacije koje menjaju kodiranje podataka (Poznati kodovi su EBCDIC, ASCII, BCD i pakovani BCD). Treba uočiti da transformacija koda ne mora biti reverzibilna. Na primer, ako je transformacija iz koda A u kod B moguća za sve elemente koda A, nije neophodno da transformacija od koda B u kod A bude moguća za sve elemente koda B. Razlog ovome može da se ogleda u činjenici da jedan od kodova ima veći broj kodnih elemenata od drugog tako da se veći broj elemenata iz jednog koda preslikava u jedan element drugog koda, dok inverzno preslikavanje ne mora bit jedinstveno.
- **Logičke operacije**. Kod ovih operacija vrednost operanda (operanada) se smatra logičkom celinom ili skup logičkih celina, a transformacije se obavljaju nad tim celinama. Osobina ovih operacija je da kada koriste podatke koji se mogu predstaviti i rezultat će imati formu koja se može predstaviti. Na primer, kod MC68020 instrukcija

  AND Src,Dst

  obavlja logičku AND operaciju nad bitovima izvornog i odredišnog operanda, a rezultat smešta na odredišnu adresu.
- **Aritmetičke operacije**. Kod ovih operacija vrednost operanda se smatra numeričkom vrednošću. Zbog toga, jedna od njihovih osobina je da se rezultat može nalaziti van opsega vrednosti koje se mogu predstaviti. Drugim rečima, rezultat može biti suviše veliki (overflow) ili suviše mali (underflow), ili da je zahtevana preciznost suviše gruba (loss of precision). Zbog toga je nakon aritmetičke operacije potrebno obaviti domen-funkciju kako bi se rezultat doveo u formu koja se može ponovo predstaviti.

Operacije za premeštanje podataka (slika 6.2) zajedno sa operacijama za transformaciju formata i koda često se zovu *operacije za manipulaciju sa podacima* (data-handling operations).



Sl. 6.2. Pregled operacija za manipulisanje sa podacima.

Svaka operacija nad podacima kao proizvod daje *rezultat* i *izvedeni rezultat*. Rezultat predstavlja prezentaciju podataka na kraju operacije. Izvedeni rezultat je informacija koja se dobija od operacije ili od rezultata operacije. Rezultati izvedeni od operacije nad podacima su zavisni od same operacije. Tipični primeri su indikatori premašaja (V), prenosa (C), nula (Z), negativan (N) koji su deo CCR-a (Condition Code Register). Drugi tip izvedenog rezultata, kao što je deljenje nulom, signalizira uslov kojim se invalidira operacija.

6.2. Operacije za premeštanje podataka

Instrukcije koje vrše premeštanje (kopiranje) podataka iziskuju specifikaciju sledećih elemenata:

- Operacija - specificirana opkodom instrukcije.
- Broj elemenata podataka - razlikujemo dve klase:
  - ♦ premeštanje samo jednog elementa (single move) - podatak koji se premešta može biti konstanta, promenljiva ili element strukture podataka (kao što je polje ili zapis).
  - ♦ premeštanje većeg broja elemenata (multiple move), javlja se u situacijama kada je potrebno premeštati celu strukturu podataka (na primer A[*], sve elemente polja A), substrukture (na primer jednu kolonu matrice B, B[I,*], ili podoblast polja A, A[9..16]). Za ovaj tip operacije potrebna je direktna hardverska podrška CPU-a.
- Lokacija izvornog operanda.
- Lokacija odredišnog operanda.

6.2.1. Premeštanje jednog podatka

Pomoću ove operacije (single data move), premešta se samo jedan elemenat. Ovo se izvodi na sledeći način:

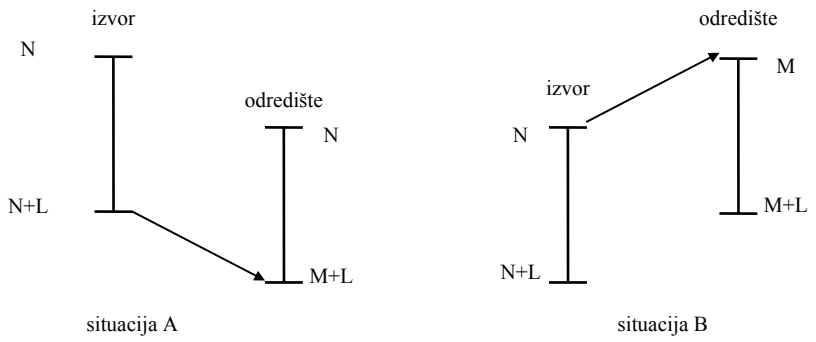
1. Premeštanje se može izvesti preko registara ili magacina.
   LOAD - vrši se premeštanje iz memorije u registar
   STORE - vrši se premeštanje iz registra u memoriju
   PUSH - vrši se premeštanje iz memorije na vrh magacina
   POP - vrši se premeštanje sa vrha magacina u memoriju
   Ovi metodi transporta interesantni su kod registarsko ili magacinsko orijentisanih arhitektura.
2. Direktno premeštanje iz memorije-u-memoriju. U ovom slučaju arhitektura mora biti sposobna da specificira dve memorijske lokacije.

6.2.2. Premeštanje većeg broja podataka

U toku ove operacije obavlja se premeštanje većeg broja elemenata. Ovo ukazuje da instrukcijom, pored lokacija izvornog i odredišnog operanda, mora da se specificira i broj elemenata, tj dužina.

Kada su i izvor i odredište u memoriji, postoji potencijalna opasnost da dođe do preklapanja izvornih i odredišnih operanada. Najjednostavnije rešenje je *destruktivno premeštanje*, kada se moguće preklapanje ignoriše. Ovo ukazuje da kompilator mora garantovati da se preklapanje neće javiti. Elegantnije, ali i kompleksnije rešenje je korišćenje *nedestruktivne* MOVE instrukcije, u toku čijeg izvršenja se prvo izvorni operand upisuje u bafer ili se sekvenca premeštanja elemenata čini zavisnom od uslova preklapanja. Na slici 6.3 je prikazan slučaj kada se u dve situacije javlja preklapanje.

Situacija A se javlja kada su izvorni podaci u odnosu na odredišne smešteni u memoriju na numerički nižim lokacijama. Kod ovakvog slučaja, moguće je realizovati nedestruktivno premeštanje startujući sa zadnjim izvornim elementom na lokaciji N+L. Kod situacije B, izvorište podataka se nalazi u memoriji sa numerički višim lokacijama u odnosu na odredište podataka, tako da prenos može da počne sa prvim izvornim elementom.



Sl. 6.3. Preklapanje izvora/odredišta.

6.2.3. Promena mesta (exchange i swap)

Ovom instrukcijom se vrši promena dva operanda. Instrukcija ovog tipa je posebno korisna kod algoritama za sortiranje. Na primer, kod MC68020 instrukcijom EXG uzajamno se menja sadržaj dva 32-bitna registra.

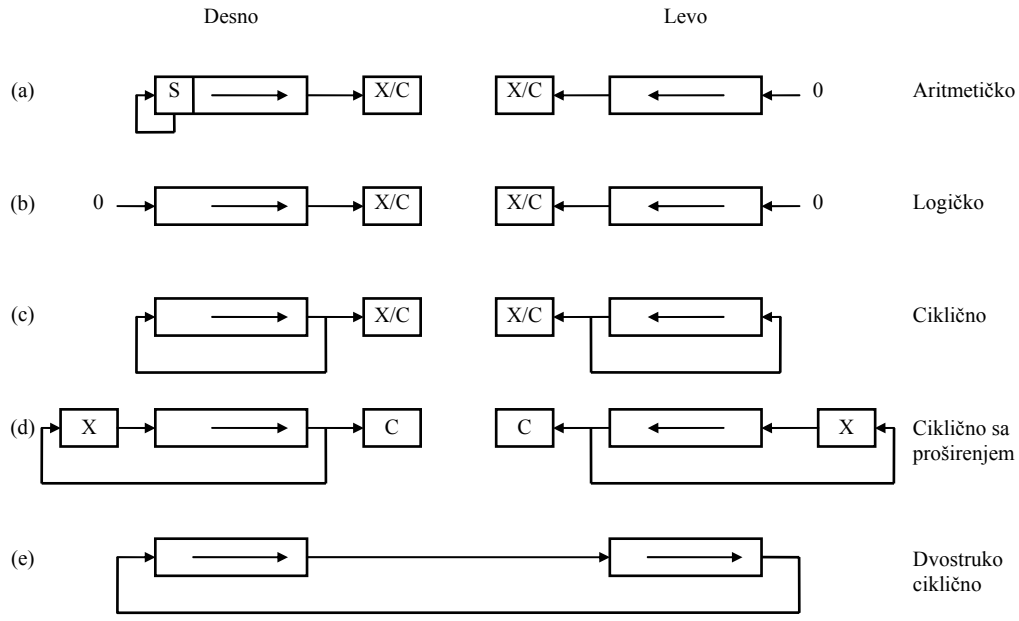
Kod MC68020 postoji takođe i poseban tip EXG instrukcije, a to je instrukcija SWAP kojom se specificira samo jedan operand. Ovo znači da se vrši premeštanje određenih delova u okviru operanda. Ova instrukcija menja obe 16-bitne polovine specificiranog registra za podatke i korisna je zbog toga što instrukcije orijentisane rečima uvek koriste LS reč 32-bitnog registra. Sa ovom instrukcijom i MS-reč se može učiniti dostupnom.

6.3. Format operacija koje vrše transformaciju

Format se specificira dodelom fiksnih lokacija kodovanoj grupi bitova. Ove operacije se standardno implementiraju kao pomeračke ili operacije koje se obavljaju nad bit poljima.

6.3.1. Operacije pomeranja

Pomeranje bitova, često kombinovano sa drugim instrukcijama, je najprostiji oblik transformacije formata. Na slici 6.4 je prikazano 5 tipova operacije pomeranja. Operacije pomeranja se obično podržavaju od strane tri klase pomeračkih instrukcija (kod koji opkod obično specificira tip operacije pomeranja):



Sl. 6.4. Tipovi operacije pomeranja.

- **aritmetičke operacije pomeranja** - aritmetičko pomeranje za N u suštini znači množenje ili deljenje broja sa N brojnih osnova (N je broj pomeranja). Ovo može da dovede do premašaja, ili da odredi odsecanje (truncation) ili zaokruživanje (rounding) rezultata. Bit znaka operanda ne učestvuje u operaciji pomeranja za slučaj kada se operacija izvodi nad brojevima u prezentaciji znak-moduo.
- **operacije logičkog pomeranja** - kod pomeranja ulevo ili udesno vrši se ubacivanje nula, a bitovi koji se pomeraju pomeraju se u X i C uslovne marker bitove.
- **operacije cikličnog pomeranja** - u ovom slučaju MSB se pomera na mesto LSB kada se vrši pomeranje ulevo, i obrnuto, kada se vrši pomeranje udesno. Neki od procesora omogućuju pomeranje ulevo/udesno nad operatorima dvostruke dužine.

Za specifikaciju operacija pomeranja važni su sledeći aspekti:

1. **tip operacije pomeranja** - aritmetička, logička ili kružna. Obično se ovo specificira opkodom.
2. **broj pomeranja** - broj pozicija koje se pomeraju. Broj pomeranja se specificira opkodom, ili u oblasti za podatke i operande. Poželjno je broj pomeranja specificirati kao konstantu ili kao promenljivu.
3. **specifikacija izvornog i odredišnog operanda** - izvorni i odredišni operand se specificiraju u prostoru za operande kada se vrši eksplicitna specifikacija. Moguće je da se jedan ili oba operanda specificiraju implicitno, ali se to mora izvesti u toku sa drugim instrukcijama.

4. **tip podataka za operand** - specifikacija o dužini operanda je često dovoljna, ali zbog nekih razloga, kao i kod MOVE operacija, veoma često se specificira i tip podataka.
5. **"single" ili "double" operand**.
6. **brojna osnova** (2 ili 10) - kod najvećeg broja slučajeva, brojna osnova je 2, sa izuzetkom kada se vrši pomeranje BCD brojeva. U ovom slučaju, brojna osnova (radix) je 10, a pomeranje se vrši u jedinicama od četiri bita.

6.3.2. Operacije sa bit poljima

Ovim operacijama kao operand se specificira bit polje. Operacije nad bit poljima se koriste kao alternativa za logička i ciklična pomeranja. One se takođe koriste za manipulaciju pakovanim podacima (koji su dozvoljeni u Pascalu).

6.4. Operacije koje vrše transformaciju koda

Kodne transformacije su veoma važne, jer veliki broj U/I uređaja koji treba da komuniciraju koriste različite kodove. Računarski sistem mora biti sposoban da manipuliše različitim kodovima i transformiše podatke iz jednog koda u drugi. Operacije koje vrše transformaciju koda se mogu klasifikovati kao tabela preslikavanja (table look-up) ili algoritamski.

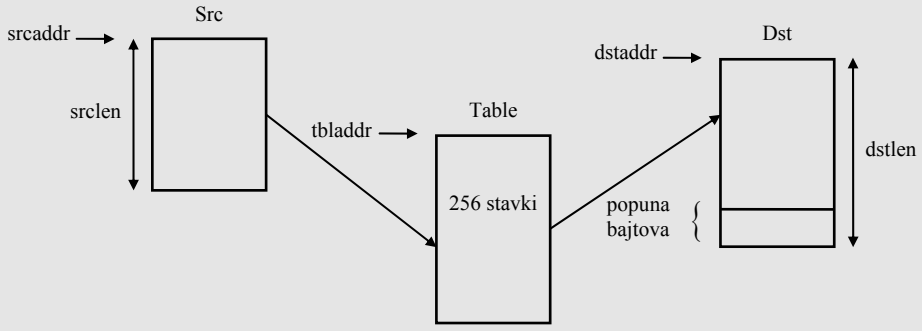
6.4.1. Transformacija koda pomoću tabela preslikavanja

Ova transformacija se koristi za jednostavnu znakovno-znakovnu translaciju, na primer za U/I uređaje. Korišćenjem tabela moguće je jednostavno manipulisati većim broj različitih skupova znakova. Veliki broj računara koristi posebne instrukcije za preslikavanje. Pored instrukcija transliranja, često skupovi naredbi sadrže i instrukcije transliraj-i-pretraži. Ove operacije pored standardnog transliranja znakova preko tabela imaju i druge funkcije. Najjednostavniji oblik ovakve operacije je onaj kod kojeg se proces translacije zaustavlja kada je rezultat znakovne translacije jednak specificiranom zanku zadatka koji se izvršava. Pomoću ove instrukcije je moguće obaviti nekoliko tipova operacija (da li elemenat pripada određenom skupu ili ne?). Tekst procesori i kompilatori mogu koristiti ovu instrukciju da bi odredili niz do delimitera (koji može biti blanko ili carriage return).

Primer 6.1:

Na slici 6.5 prikazana je MOVTC instrukcija (VAX-11), koja ima šest operanada

MOVTC srclen,srcaddr,fill,tbladdr,dstlen,dstaddr



Sl. 6.5. Instrukcija MOVTC kod mašine VAX-11.

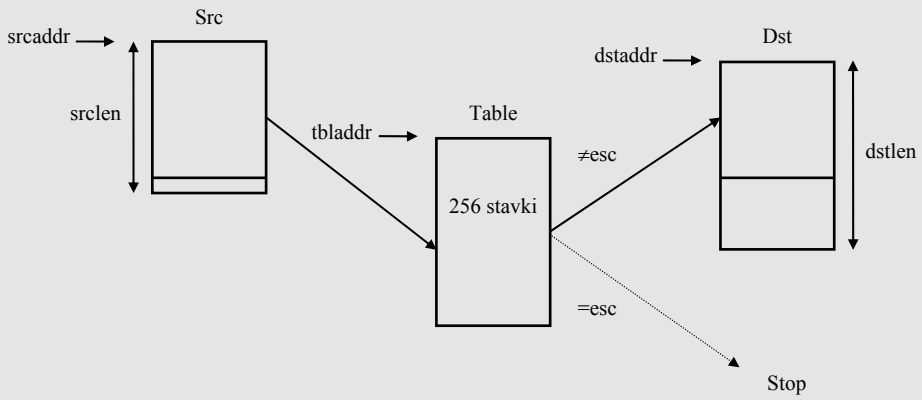
Operand fill se koristi za popunu i čini da instrukcioni znakovni niz bude nezavisan. Znaci izvorišta specificirani izvornom dužinom i adresom operanada se prevode, a rezultat direktno smešta u odredište, koje se takođe specificira svojom dužinom i adresom operanda. Prevođenje se izvodi pomoću tabele koju čini 256 ulaza. Tabela je smeštena u memoriji na lokaciji specificiranoj operandom adresa tabele. Svaki znak iz izvorišta se koristi kao indeks u ovoj tabeli, a izabrana vrednost se memoriše u odgovarajuću odredišnu lokaciju.

Primer 6.2:

Na slici 6.6 je prikazana VAX-11 instrukcija MOVTUC koja ima šest operanada

MOVTUC srclen,srcaddr,esc,tbladdr,dstlen,dstaddr

Operand esc je test znak. Transliranje se zaustavlja kada je rezultat znakovne translacije jednak ovom znaku.



Sl. 6.6. Instrukcija MOVTUC kod VAX-11.

6.4.2. Algoritamske kodne transformacije

Sa izuzetkom instrukcija opšte namene, koje manipulišu znakovima (uglavnom se koriste za U/I aktivnosti i da kompilator odredi delimitere), postoje i instrukcije za kodnu transformaciju koje se odnose na internu prezentaciju. Tipičan primer je translacija između decimalne i binarne prezentacije, između brojeva u fiksno i pokretnom zarezu, i između znakova, sa jedne, i brojeva u fiksnom i pokretnom zarezu, sa druge strane.

6.5. Logičke operacije

Kod najvećeg broja programa logičke operacije se koriste sa ciljem da se donese neka odluka. Formalno, logičke operacije se definišu samo na jedinstvenim bitovima, ali se kod računara one izvršavaju nad niskom bitova ili bit vektorima. U ovakvim situacijama specificirana logička operacija se izvršava po principu bit-sa-bitom (bit-wise) nad svim bitovima operanda, tako da je ona u suštini vektorska operacija.

Logičke vektor operacije (kada je vektor fiksne dužine) se lako mogu realizovati, pošto su elementi vektora bitovi. Najveći broj arhitektura ima četiri logičke operacije: AND, OR, EOR i NOT. Često se umesto EOR koristi akronim XOR.

Izvedeni rezultati logičkih operacija se generišu kao sporedni efekti (side effect) ovih operacija. U opštem slučaju, logičke operacije generišu rezultantni vektor od koga se mogu izvesti određene osobine. Ove osobine mogu biti, na primer, signaliziranje sve-nula (all zero) rezultata, i brojanjem bitova jednakih 0 ili 1. Ovi rezultati se mogu generisati direktno kao sporedni efekat operacije i čuvati u specijalnom registru (CCR), ili da se odrede pomoću specijalnih instrukcija.

Primer 6.3:

U Tabeli 6.1 su prikazane logičke operacije mikroprocesora MC68020. Uočimo da instrukcija tipa EOR <ea>,Dn nije implementirana.

Tab. 6.1. Logičke instrukcije kod MC68020.

Instrukcija	Sintaksa operanda	Efekat
AND	Src,Dn Dn,Dst	Dst:=(Dn) AND (Src) Dst:=(Dst) AND (Dn)
ANDI	#data,Dst	Dst:=(Dst) AND data
EOR	Dn,Dst	Dst:=(Dst) EOR (Dn)
EORI	#data,Dst	Dst:=(Dst) EOR data
NOT	Dst	Dst:=NOT (Dst)
OR	Src,Dn Dn,Dst	Dn:=(Dn) OR (Src) Dst:=(Dst) OR (Dn)
ORI	#data,Dst	Dst:=(Dst) OR data

Dn je registar za podatke; (Src), (Dst) predstavlja sadržaj lokacija sa navedenim efektivnim adresama; #<data> je neposredni podatak.

6.6. Aritmetičke operacije u fiksnom zarezu

Aritmetičke operacije (kao što su +, -, * i /) se mogu izvršavati nad brojevima u FixP (Fixed Point) ili nad brojevima u FP (Floating Point), kao i nad brojevima FL (fixed length - fiksne dužine) ili VL (variable length - promenljive dužine), nad brojnom osnovom binarnom ili decimalnom.

Aritmetčke operacije se izvršavaju nad brojevima koji se mogu predstaviti, a takođe i rezultat treba da ima oblik koji se može predstaviti. Imajući ovo u vidu, javljaju se dva domenska problema:

1. Rezultat može biti van opsega koji se može predstaviti. U tom slučaju funkcija "promene opsega" mora da promeni vrednost. Kada je vrednost rezultata suviše velika u odnosu na opseg prestavljanja, javlja se premašaj, a kada je vrednost suviše mala javlja se podbačaj (ovo se javlja kod FP). Premašajem i podbačajem se može manipulisati odsecanjem (truncating) MS bitova i signaliziranjem (postavljanjem OF bita) ili memorisanjem MS bitova zajedno sa bitovima za signalizaciju.
2. Rezultat se ne može predstaviti, zbog odsecanja LS cifara. Na primer, 1/3=0.3333... U ovom slučaju, kažemo da se radi o ograničenoj preciznosti kod predstavljanja brojeva.

Tretman preciznosti ukazuje na izbor zadovoljavajuće vrednosti pomoću koje vršimo predstavljanje broja koji se ne može predstaviti. Najpoznatiji metodi su *odsecanje* (truncating) i *zaokruživanje* (rounding). Rezultat odsecanja zavisi od predstavljanja brojeva (Tabela 2). Odsecanje ima simetričan efekat prema nuli kod prezentacije znak-moduo, dok je efekat odsecanja kod cifarskog komplementa i komplementa brojne osnove prema negativnim brojevima. Rezultat zaokruživanja je identičan za sve prezentacije, pa shodno tome i više se izvodi (preferira se). Zaokruživanje je, na žalost, nešto teže za implementaciju zbog obaveznog sabiranja konstantne vrednosti.

Tretman preciznosti se može implicitno obaviti (na primer, kao sporedni efekat aritmetičke operacije) ili eksplicitno, korišćenjem specijalnih instrukcija. Kod implicitnog pristupa suvišne (superfluous) cifre se izbacuju nakon operacije, dok one moraju ostati dostupne kod tretmana tipa eksplicitne preciznosti. Neke arhitekture imaju tzv. mode-bit koji određuje kada se izvodi implicitno odsecanje ili zaokruživanje.

Tab. 6.2. Efekti tretmana preciznosti.

Tretman preciznosti	Reprezentacija	Rezultat								
		-1		-0.5		0		+0.5		+1
Odsecanje	Znak veličine	-1	→	-0.5	→	0	←	+0.5	←	+1
Odsecanje	Komplement osnove	-1	←	-0.5	←	0	←	+0.5	←	+1
Odsecanje	Komplement najveće cifre	-1	←	-0.5	←	0	←	+0.5	←	+1
Zaokruživanje	Sve notacije	-1	←	-0.5	→	0	←	+0.5	→	+1

6.6.1. Aritmetika nad podacima fiksne dužine.

Karakteristična razlika između aritmetičkih operacija nad brojevima u FixP i matematičke aritmetike (aritmetike na konceptualnom nivou) ogleda se kod predstavljanja brojeva. Ovo znači da se dodatne mere moraju učiniti kako bi se indiciralo kada preciznost nije dovoljno dobra, i kada je moguće poboljšati (povećati) tu preciznost. Shodno tome, mogućnost za povećavanje preciznosti se mora razmatrati kod svake operacije. U daljem tekstu diskutovaćemo izvođenje operacija +, -, * i / u kontekstu povećane preciznosti.

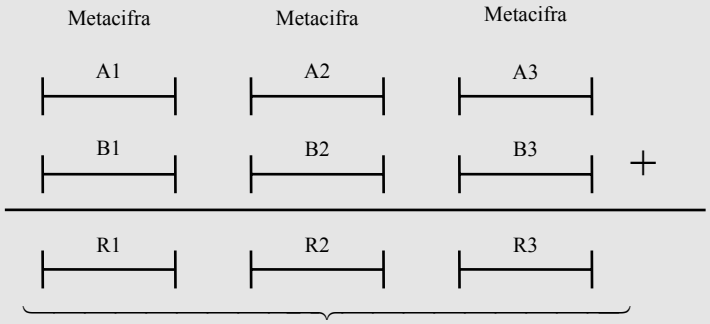
Sabiranje i oduzimanje

Operacije "+" i "-" smatraju se simetričnim, sa izuzetkom oduzimanja broja od broja nula u notaciji komplement-brojne-osnove. Kod ovog slučaja javlja se premašaj, tj. $0-(-2^{N-1})$ uslovljava premašaj. Kao sporedni efekat ovih operacija, informacija o rezultatu se obično ostvaruje korišćenjem indikacije tipa "nula", "negativan" ili "pozitivan" koji su deo CCR-a. Da bi omogućili realizaciju veće preciznosti sa instrukcijama iz grupe jednostruke preciznosti uzima se u obzir signal carry/borrow koji je deo CCR-a.

Povećana preciznost (*extended precision*) kod "+" i "-" obično ukazuje da se izvode operacije nad onom prezentacijom podataka koji su umnožak standardnog načina prezentacije. Na primer, ako je standardni tip podataka 16-bitna celobrojna vrednost, operacije povećane preciznosti su operacije nad celobrojnim vrednostima dužine 32, 48, 64,... bitova. Programi i potprogrami pomoću kojih se ostvaruju ove funkcije vide brojeve povećane preciznosti kao metabrojeve. Metabrojevi se grade od brojeva fiksnih dužina koje cifre metabroja (metacifre) u notaciji povećane preciznosti. Brojna osnova ove meta-notacije je moduo prezentacije metacifara, tj. ako su metacifre 16-bitne celobrojne vrednosti, brojna osnova metabroja je 2^{16}. Operacije "+" ili "-" ostvaruju se korišćenjem metacifara i bita prenos/pozajmica (*carry*/*borrow*).

Primer 6.4:

Neka su A, B i R 48-bitni metabrojevi, svaki sastavljen od tri 16-bitne metacifre. Sabiraju se A i B a rezultat dodeljuje R-u (slika 6.7). Sabiranje se sastoji od jedne "regularne" operacije "+" i dve operacije sabiranja sa prenosom (da bi se ostvarila povećana preciznost) na metacifarskom nivou.




```
R3:=A3+B3       ; izvedeni rezultat: Carry:=0 ili 1
R2:=A2+B2+C     ; izvedeni rezultat: Carry:=0 ili 1
R3:=A1+B1+C     ; izvedeni rezultat: Carry:=0 ili 1
```

Sl. 6.7. Sabiranje dva metabroja.

Množenje

Kod operacija "+" i "-" rezultat je jednostruke dužine (*single length*) sa dodatkom indikacije mogućeg premašaja. Kod operacije "*" generiše se rezultat dvostruke dužine (*double length*), koji se može koristiti za operacije povećane preciznosti. Sa aspekta HLL, poželjno je da tip podataka proizvoda bude isti kao i tip oba operanda, tj. proizvod dve 16-bitne celobrojne vrednosti I1 i I2 bude ponovo 16-bitna celobrojna vrednost, što će omogućiti realizaciju iskaza dodele, kao što je I1:=I1*I2. Kod najvećeg broja računara moguće je praviti izbor između jednostruke ili dvostruke dužine rezultata. Koja se od njih želi specificira se u opkod prostoru, prostoru operanada ili prostoru za podatke (koristeći tagove). Ako je izabran opkod, potrebno je izvesti dve operacije množenja: jednu kojom se generiše proizvod jednostruke preciznosti (LS deo), a jednu kojom se generiše kompletan proizvod dvostruke dužine, ili samo MS deo.

Opcija specifikacija dužine rezultata u prostoru operanada koristi fakat da su neki operandi implicitno označeni kao operandi dvostruke dužine, pri čemu isti koriste dve uzastopne odredišne lokacije, pri čemu se specificira samo jedna od njih. Obično, za ove lokacije se koristi registarski par. Par se specificira na sledeći način:

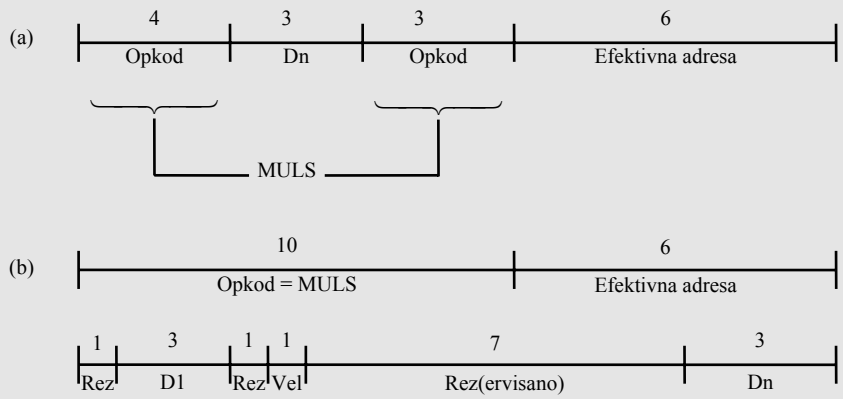
1. Registar R i (R OR 1); gde R OR 1 znači da od bit oblika koji specificira R (na primer, 010 za R2), LS bit je postavljen. Tako da, ako R=010, tada R OR 1=011, što znači da će se koristiti R2 i R3. Ako je R=011, tada će se koristiti samo R3 za oba rezultata, što znači da će biti dostupan samo zadnje zapamćeni rezultat. Ako je R paran broj, tada će se koristiti sledeći registar, a ako ne samo označeni registar.
2. Registar R i (R OR 1).
3. Registar R i (R + 1).

Rešenja (2) i (3) uvek ukazuju na rezultat dvostruke dužine, a rešenje (3) zahteva dodatno sabiranje da bi se identifikovao drugi registar.

Primer 6.5:

Kod MC68020 postoje dva tipa instrukcije množenja, jedna za označeno množenje (MULS) a druga za neoznačeno množenje (MULU). Operacija neoznačenog množenja je važna kod operacija povećane preciznosti, jer samo MS metacifra ima znak. MC68020 podržava operacije MULS i MULU pomoću tri instrukcije. Na slici 6.8 prikazana je samo MULS. Instrukcije (1) i (3) se mogu koristiti kod operacija povećane preciznosti.

```
(1)     MULS.W      <ea>,Dn         16*16→32    ; dvooperandski format, slika 6.8a
(2)     MULS.L      <ea>,D1         32*32→32    ; dvooperandski format, slika 6.8a
(3)     MULS.L      <ea>,Dh:D1      32*32→64    ; dvooperandski format, slika 6.8b
```



Sl. 6.8. Format MUL instrukcije kod MC68020.

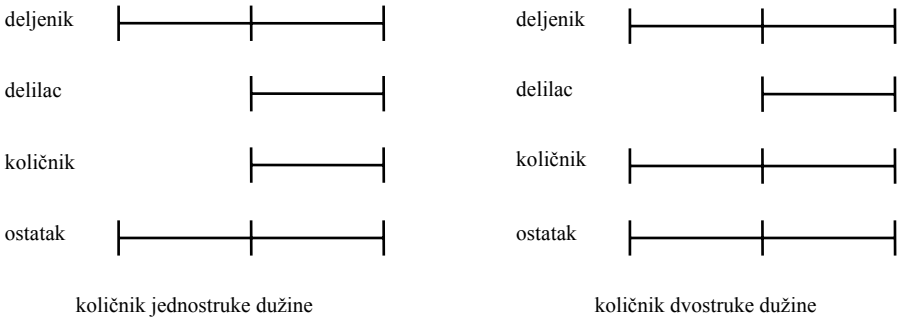
Instrukcije MULS i MULU mogu imati dva ili tri operanda. Instrukcija (1) koristi dvooperandski format (slika 6.8a), a generiše rezultat dvostruke dužine. Smanjenje dužine proizvoda je beskorisno jer su svi registri za čuvanje podataka kod MC68020 32-bitni. 16-bitni rezultat se može dobiti od LS reči rezultantnog registra. Instrukcija (2) može koristiti dvooperandski format. Instrukcija (3) koristi trooperandski format (slika 6.8b). Operacija se realizuje na sledeći način: LS 32 bita proizvoda od (<ea>) i (Dl) se smeštaju u Dl dok se MS 32 bita smeštaju u Dh.

Deljenje

Kao i kod množenja, rezultat deljenja se ne može uvek predstaviti, na primer, 1/3=0.333... . Da bi učinili deljenje tačnim, može se generisati i ostatak, pomoću koga se ukazuje da operacija deljenja generiše dva rezultata. Kako je ostatak uvek manji od delioca, dužina ostatka može biti jednaka sa dužinom delioca.

Kada se deljenje smatra inverznom operacijom množenja, deljenik može biti broj jednostruke ili dvostruke dužine (slika 6.9.).

Kod najčešće korišćenih implementacija se usvaja da je količnik jednostruke dužine. Kod HLL-ova se usvaja da su oba broja (deljenik i delilac) jednostruke dužine; na primer, I1:=I1/I2, gde su I1 i I2 16-bitne celobrojne vrednosti.



Sl. 6.9. Dužina količnika kod operacije deljenja.

Tab. 6.3. Varijacije znaka količnika i ostatka.

		Ojlerova celobrojna aritmetika		Modulus aritmetika	
Deljenik	Delilac	/	REM	DIV	MOD
7	3	2	1	2	1
7	-3	-2	1	-3	-2
-7	3	-2	-1	-3	2
-7	-3	2	-1	2	-1

Shodno Tab. 6.3, operacija deljenja se može realizovati na dva načina:

(1) Koristeći Ojlerovu celobrojnu aritmetiku: Za količnik i ostatak se koriste operatori / i REM. Znak ostatka je isti kao i deljenika. Neka količnik bude $q=m/n$,a ostatak $r=m$ REM n. Tada važe sledeće relacije:
$$q*n+r=m \text{ i } 0\leq ABS(r)<ABS(n)$$
(2) Koristeći Modulus aritmetiku: Za količnik i ostatak se koriste operatori MOD i DIV. Neka je $Q=m$ DIV n i $R=m$ MOD n, gde je znak ostatka jednak znaku delitelja.

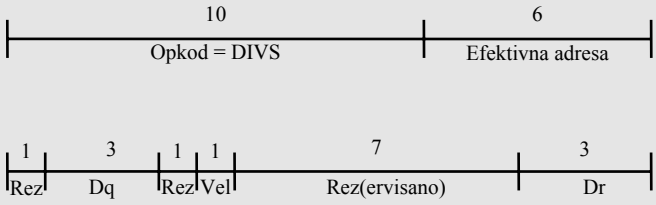
Kod najvećeg broja jezika, kao što su Pascal i C, koristi se Ojlerova celobrojna aritmetika, koja je podržavana od najvećeg broja arhitektura (MC68020, VAX-11). Neke arhitekture, kao što je NS32000, imaju posebne instrukcije za generisanje količnika i ostatka.

Primer 6.6:

Kod MC68020 postoje instrukcije za označeno deljenje (DIVS i DIVSL) i neoznačeno deljenje (DIVU i DIVUL). Operand specificiran efektivnom adresom je uvek delitelj, dok se u odredišnom registru (registrima) čuva deljenik. Samo jedna instrukcija deljenja operiše nad rečima a ostale tri nad dugim rečima.

(1)	DIVS.W	<ea>,Dn	32/16→16r:16q	; dvooperandski format
(2)	DIVS.L	<ea>,Dq	32/32→32q	; dvooperandski format
(3)	DIVS.L	<ea>,Dr:Dq	64/32→32r:32q	; trooperandski format
(4)	DIVS.L	<ea>,Dr:Dq	32/32→32r:32q	; trooperandski format

Instrukcija (1) generiše količnik u LS reč registra Dn, a ostatak u MS reč Dn-a. Instrukcija (3) usvaja da je reč deljenika četvorostruka i nalazi se u registarskom paru Dr:Dq, a nakon izvršenja instrukcije 32-bitni ostatak je u Dr, a 32-bitni količnik u Dq. Dvooperandski format DIVU i DIVS instrukcija je identičan sa MULU i MULS instrukcijama, (slika 6.8a). Trooperandski format DIV instrukcije prikazan je na slici 6.10.



Sl. 6.10. Trooperandski format DIV instrukcije kod MC68020.

6.6.2. Aritmetika nad podacima promenljive dužine

Operacije sa celobrojnim vrednostima promenljive dužine su skoro identične sa onim koje se odnose na fiksne dužine. Razlika je u tome što se mora specificirati dužina operanda.

Najvažnija oblast primene podataka promenljive dužine je administracija, zbog važnosti brojeva u fiksnom zarezu promenljive dužine. Zbog toga, najveći broj mašina koje nude aritmetiku promenljive dužine ograničava se na korišćenje brojeva u BCD prezentaciji.

Dužina operanda se može specificirati na sledeći način:

- **u opkod prostoru**: ovo nije efikasno zbog potencijalno velikog broja instrukcija (za svaku dužinu opkod treba da je različit).
- **u prostoru operanada**:
  - specificiranjem dužine kao neposredne vrednosti (IBM/370).
  - specificiranjem dužine preko posebnog operanda (VAX-11).
- **u prostoru podataka**:
  - dodavanjem podatku polja koje ukazuje na dužinu (deskriptor).
  - dodavanjem specijalnog bit oblika (flag) kojim se markira kraj broja. Ovo se lako izvodi kada se koristi BCD kod, jer 10 od 16 mogućih kodova predstavljaju decimalni broj.

6.7. Aritmetičke operacije u pokretnom zarezu

FP brojevi se predstavljaju pomoću eksponenta i mantise (uključujući znak), tj. njihova vrednost je određena na osnovu izraza

$$\text{vrednost}=(-1)^{\text{znak}} * \text{mantisa} * \text{brojna\_osnova}^{\text{eksponent}}$$

Operacije koje se obavljaju sa FP brojevima mogu se uporediti sa onim u FixP aritmetici, sa razlikom što, u zavisnosti od operacije, određeni matematički zakon važi samo aproksimativno. Na primer, asocijativni zakon A+(B+C)=(A+B)+C važi samo aproksimativno zbog relativne greške koja normalno prati izvođenje FP operacija. Ova greška se može učiniti manjom od:

brojna_osnovaeksponent *0.5*brojna_osnova(1-F)

gde se F odnosi na broj cifara mantise za normalizovane brojeve (koristeći zaokruživanje). Za slučaj da postoji odsecanje, ova greška je dva puta veća. Da bi smanjili grešku. kod velikog broja mašina se koriste FP brojevi u dvostrukoj preciznosti koji imaju format sa dva puta većim brojem bitova u odnosu na jednostruku tačnost. Ovi dodatni bitovi se mogu koristiti da prošire polje mantise (povećana preciznost), a kod nekih arhitektura i za polje eksponenta (povećani opseg). Preciznost FP broja je maksimalna ako je broj normalizovan. Zbog toga, kod najvećeg broja arhitektura se koriste normalizvani operandi, a i generisani rezultat je normalizovan.

Sabiranje i oduzimanje

Kako se FP operandi sastoje iz dva dela, eksponenta i mantise, sabiranje i oduzimanje se obavljaju u saglasnosti sa algoritmom koji uzima u obzir oba dela. Operacije sabiranje i oduzimanje se često prevode u operacije tipa efektivno sabiranje i oduzimanje, saglasno Tabeli 6.4.

Tab. 6.4. Operacije sabiranje/oduzimanje u notaciji znak/moduo.

Željena operacija	Znak Op1	Znak Op2	Operacija koja se izvodi	Efektivna operacija
ADD.F	+	+	Op1 + Op2	Sabiranje
ADD.F	+	-	Op1 - OP2	Oduzimanje
ADD.F	-	+	Op2 - Op1	Oduzimanje
ADD.F	-	-	-(Op1 + Op2)	Sabiranje
SUB.F	+	+	Op1 - Op2	Oduzimanje
SUB.F	+	-	Op1 + Op2	Sabiranje
SUB.F	-	+	-(Op1 + Op2)	Sabiranje
SUB.F	-	-	Op2 - Op1	Oduzimanje

Algoritmi za efektivno sabiranje

Efektivno sabiranje znači da operandi imaju isti znak. Sabiranje i oduzimanje se može ostvariti samo kada su eksponenti jednaki. Alogoritam za efektivno sabiranje čine sledeća četiri dela:

1. Deo kojim se proverava kada je poravnanje neophodno. Poravnanje je neophodno kada eksponenti nisu jednaki:

```
if Exp1=Exp2 then
        begin (*Poravnanje nije neophodno*)
                ExpRez:=Exp1;
                goto c; (*Idi na korak (3)*)
        end;
if ABS(Exp1-Exp2)>Broj_cifara_mantise then
        begin (*Grubo poravnanje*)
                if Exp1>Exp2 then
                        Rez:=Op1
                else
                        Rez:=Op2;
                        goto Uradi;
        end;
```

2. Deo za poravnanje mantisa

```
if Exp1>Exp2 then
        begin
                shiftright Fraction2,(Exp1-Exp2) cifara;
                ExpRez:=Exp1;
        end;
else
        begin
                shiftright Fraction1,(Exp-Exp2) cifara;
                ExpRez:=Exp2;
        end;
```

3. deo za sabiranje mantisa

```
FractionRez:=Fraction1+Fraction2;
```

4. Deo koji proverava normalizaciju rezultata i normalizuje ga kada je to neophodno

```
if fraction_premasaj then
        begin (*Post-normalizacija 1 cifre*)
                shiftright FractionRez,1 cifra;
                if exp_premasaj then
                        postavi indikator_premasaja;
                end;
```

Uradi:

Algoritam za efektivno oduzimanje

Algoritam za efektivno oduzimanje (sabiranje operanda sa različitim znakom) razlikuje se od efektivnog sabiranja samo u delovima (3) i (4). Deo (3) se razlikuje, jer se mantise oduzimaju. Deo (4), normalizacioni korak, je različit jer, zbog operacije oduzimanja u delu (3), ne postoji premašaj u mantisi. U ovom slučaju, ipak, moguće je da postoji broj kod koga su cifre veće težine nula zbog operacije oduzimanja. Zbog toga deo (4) mora da se promeni i njegov oblik je sledeći:

```
if cifra_mantise_najvece_tezine=0 then
        begin (*Normalizacija*)
                if FractionRez:=0 then
                        begin (*Specijalni slucaj kada je rezultat 0*)
                                FractionRez:=0;
                                ExpRez:=0;
                                goto Uradi;
                        end;
                shiftright FractionRez,broj_0_cifara;
                ExpRez:=ExpRez-broj_0_cifara;
                if Exp_podbacaj then
                        postavi indikator_podbacaja;
                end;
```

Uradi:

U ovom slučaju je izvedena specijalna provera da proveri kada je rezultat postao nula.

Zaštitni bitovi

Poželjno je da rezultat operacije aproksimira egzaktni rezultat što je moguće tačnije. Zbog toga je važno da relativna greška bude manja od vrednosti LS cifre normalizovanog rezultata. Zbog ovoga se uvode zaštitni (guard) bitovi, Zaštitni bitovi su cifre desno od značajnih cifara mantise (one koje se predstavljaju) a njihova uloga je da obezbede zaštitu od gubitka cifara koje su kasnije levo u frakciji u toku normalizacije. Primer ovoga je operacija A-

B (koristi brojnu osnovu 10). Efektivno oduzimanje koje se izvodi bez korišćenja zaštitne cifre usloviće da se LS cifra izgubi, kao što je prikazano sledećom tabelom.

Eksponent	Mantisa	Poravnanje	Zaštitna cifra	Operacija
2	1.69	$10^2 * 1.69 \rightarrow 10^2 * 1.69$	0	A
1	7.85	$10^1 * 7.85 \rightarrow 10^2 * 0.78$	5	B
1	9.05	$10^1 * 9.05 \rightarrow 10^2 * 0.90$	5	A - B

Pokažimo sada, da je u toku operacija sabiranja i oduzimanja potreban samo jedan zaštitni bit, da bi se dobila dodatna preciznost.

Sabiranje - Dva broja se sabiraju; jedan se podešava, a drugi ne mora

$a_0\ a_1\ \ldots\ a_{p-1}\ 0\ 0$ broj sa većim eksponentom
$0\ 0\ b_0\ \ldots\ b_{p-3}\ b_{p-2}\ b_{p-1}$ + broj sa manjim eksponentom

$c\ r_0\ r_1\ \ldots\ r_{p-1}\ r_p\ r_{p+1}$

U svim slučajevima 'cr_0' $\geq Q_0 = 1$ (bit prenosa c i bit r_0), tako da normalizacija na levo nije potrebna pa zbog toga i zaštitna cifra nije potrebna. Jedna zaštitna cifra se može koristiti za zaokruživanje.

Oduzimanje - moguće je razlikovati tri slučaja

(1) podešavanje nije neophodno (jednaki eksponenti)

$a_0\ a_1\ \ldots\ a_{p-1}\ 0$
$b_0\ b_1\ \ldots\ b_{p-1}\ 0$ -

$b\ r_0\ r_1\ \ldots\ r_{p-1}$

Kada su eksponenti jednaki, normalizacija i zaštitna cifra nisu potrebne.

(2) podešavanje=1 (eksponenti se razlikuju za 1)

$a_0\ a_1\ \ldots\ a_{p-1}\ 0$ broj sa većim eksponentom
$0\ b_0\ b_1\ \ldots\ b_{p-2}\ b_{p-1}$ - broj sa manjim eksponentom

$r_0\ r_1\ \ldots\ r_{p-1}\ r_p$

r_p može biti različito od nule. Normalizacija ulevo za jednu ili veći broj cifarskih pozicija je neophodna ako je $r_0 = 0$ - tj. ako $a_0 = 1$ i postoji pozajmica od a_0 - tako da je dovoljan samo jedan zaštitni bit, jer, zbog toga što je poravnanje=1, više zaštitnih cifara i ne postoji.

(3) poravnanje≥2 (eksponent se razlikuje za dva ili više)

$a_0\ a_1\ a_2\ \ldots\ a_{p-1}\ 0\ 0$ broj sa većim eksponentom
$0\ 0\ b_0\ \ldots\ b_{p-3}\ b_{p-2}\ b_{p-1}$ - broj sa manjim eksponentom

$r_0\ r_1\ \ldots\ r_{p-1}\ r_p\ r_{p+1}$

Normalizacija ulevo za dve ili veći broj cifarskih pozicija je potrebna samo ako je $r_0 = r_1 = 0$. Uslov $r_0 = 0$ ukazuje da $a_0 = 1$ i pozajmljivanje od a_0 je zahtevano od strane a_1. Čak a_1 pozajmljuje od a_0 samo ako $a_1 = 0$ i a_2 pozajmljuje od a_1. Ako $a_1 = 0$, ipak i a_2 pozajmljuje od a_1, tada $r_1 \neq 0$. Zbog ovoga normalizacija je neophodna samo za jednu cifru, tako da je dovoljna samo jedna zaštitna cifra.

Množenje i deljenje

Obe ove operacije očekuju operande određenog formata i generišu rezultat u istom formatu. Ovo znači da, u principu, množenje generiše frakciju duple dužine. Shodno tome, preciznost operacije koja se primenjuje na tu mantisu mora biti takva da se mora uklopiti u korektni format. Operacija kojom se određuje preciznost mora biti

tipa odsecanje ili zaokruživanje. Ako usvojimo da su operandi predstavljeni u normalizovanoj formi, algoritam će biti sledeći

Množenje	Deljenje
(1) sabiranje eksponenata	(1) oduzimanje eksponenata
(2) množenje mantisa	(2) deljenje mantisa
(3) normalizacija rezultata	(3) normalizacija rezultata
(4) tretman preciznosti (zaokruživanje ili odsecanje)	(4) tretman preciznosti

Povećana preciznost

Povećanu preciznost kod FP je teško ostvariti sa FP instrukcijama za jednostruku preciznost. Ovo je posledica činjenice da su ove standardne instrukcije tako projektovane da se vrši normalizacija operanada i rezultata. Mada FP metabrojevi treba da se normalizuju, svaka posebna metacifra ne mora. Ako se pokuša realizacija povećane preciznosti korišćenjem normalnih instrukcija za jednostruku preciznost, dovći će do mogućeg premašaja ili podbačaja u polju eksponenta kod svake cifre. Šta više, operacije nad metaciframa zahtevaju podešavanje operanda.

Argumenti pokazuju da FP brojevi jednostruke preciznosti nisu najpogodniji kao metacifre za povećanu preciznost FP brojeva. Brojevi sa fiksnom dužinom i FixP operacije su pogodniji za povećanu preciznost, kod FP aritmetike. Veliki broj arhitektura omogućava izvođenje operacija tipa duple preciznosti.

Ekstremumi

Kod rada sa FP brojevima javljaju se krajnosti koje se odnose na opseg i preciznost FP brojeva.

Ekstremumi opsega

Odnose se na maksimalnu i minimalnu vrednost eksponenta. Iznad maksimalne se javlja premašaj, a ispod minimalne podbačaj. Greška premašaja, koja se obično signalizira preko trapova (softverskih prekida), ukazuje da je FP broj suviše mali i da treba koristiti povećanu preciznost. Greška podbačaja ukazuje da je rezultat suviše mali. Akcije koje se preduzimaju svode se na dodelu nule rezultatu ili signaliziranje ove situacije. Signaliziranje podbačaja i premašaja treba tako da se izvede da se rekonstruišu korektne vrednosti. Rešenja koja generišu nepredvidljivi (nedefinisani) rezultat nisu poželjna, jer čine da se operacija korekcije ne može izvesti. Signaliziranje se izvodi na dva načina:

(1) Postavljanjem markera (na primer V za premašaj a U za podbačaj) CCR-a, što će kasnije omogućiti programeru da ih testira.
(2) Generisanjem trapa, koji se kod nekih arhitektura može maskirati. Kod trap maskirajućeg mehanizma, selektivne reakcije na automatsku detekciju premašaj/podbačaj su moguće.

Ekstrmumi preciznosti

Ove krajnosti se odnose na maksimalni i minimalni broj značajnih cifara mantise. Uobičajene metode za manipulaciju sa značajnim ciframa su odsecanje i zaokruživanje. Kod velikog broja arhitektura postoji "mode bit" kojim se bira željeni metod.

Drugi važan aspekt koga treba razmotriti odnosi se na minimalni broj značajnih cifara mantise. Ovaj broj se može u velikoj meri redukovati efektivnim oduzimanjem približno dva identična broja. Rezultat efektivnog oduzimanja:

$$R = 0.14749588E+5 - 0.14769578E+5 =$$
$$= 0.00000010E+5 = 0.100000000E-1$$

daje nekorektnu ideju da je normalizovani rezultat određen sa devet značajnih cifara, a u suštini samo su dve cifre značajne. Kod IEEE FP standarda postoji posebno kodiranje za veoma male brojeve (oni koji su bliski nuli). Ovo kodiranje se zove denormalizacija brojeva. U ovom slučaju je eksponent=0, a mantisa$\neq$0.