3. STRUKTURE PODATAKA

Osnovni ili primitivni tipovi podataka sa kojima se instrukcijama mikroprocesora MC68020 direktno operiše, su označene i neoznačene celobrojne vrednosti, BCD celobrojne vrednosti i Boolean promenljive. Nasuprot njima, nizovi podataka se moraju kreirati a njima se manipuliše algoritmima koji su projektovani od strane programera. Definicija novih tipova podataka (koji nisu dostupni na nivou asemblera), i njihov logički uzajamni odnos određuje njihovu organizaciju, ali takođe ukazuje da se radi o strukturama podataka.

Polje je primer strukture podataka. Polja se sastoje od skupa stavki jedinstvenih tipova podataka smeštenih u kontinualnim memorijskim lokacijama. Alternativni termin za "polje" je "tabela".

Povezane liste su drugi tip strukture podataka koje se koriste kod velokog broja aplikacija. Struktura omogućava da se stavke podataka smeštaju u nekontinualne memorijske lokacije koristeći se pokazivačima koji ukazuju na lokaciju sledeće stavke u listi.

3.1. Jednodimenzionalna polja

Jednodimenzionalno polje je struktura koju čini skup stavki kod kojih se svaki elemenat jedinstveno identifikuje vrednošću koja odgovara njegovoj poziciji u polju. Imajući u vidu da je svaka stavka polja istog tipa za strukturu kažemo da je homogena. U matematici, jednodimenzionalno polje brojeva se zove vektor, pri čemu se pozicija svakog elementa specificira indeksom, kao što je

$$v_1, v_2, \ldots, v_N$$

za vektor od N elemenata.

Kod FORTRAN-a prvi elemenat vektora ima indeks 1, kao što je

$$V(1), V(2), \ldots, V(N)$$

gde V(i) ukazuje na adresu elementa V_i. Kod Pascala i Algola dozvoljena je proizvoljna specifikacija prvog indeksa. Kod asemblerskog jezika dozvoljena je kompletna fleksibilnost.

Sekvencijalno memorisanje elemenata jednodimenzionalnog polja dozvoljava da se svakom elementu vrši obraćanje shodno njegovoj adresi. Ako polje X od N elemenata počinje sa lokacije X(1), a svaki elemanat zauzima memorijski prostor od C bajtova, tada je adresa j-tog elementa

$$X(j)=X(1)+C*(j-1), \quad 1 \le j \le N$$

gde je C konstanta. Dužina polja u bajtovima je

$$\text{dužina}=(X(N)-X(1))*C$$

gde X(N) i X(1) predstavljaju adrese prvog i zadnjeg elementa, respektivno.

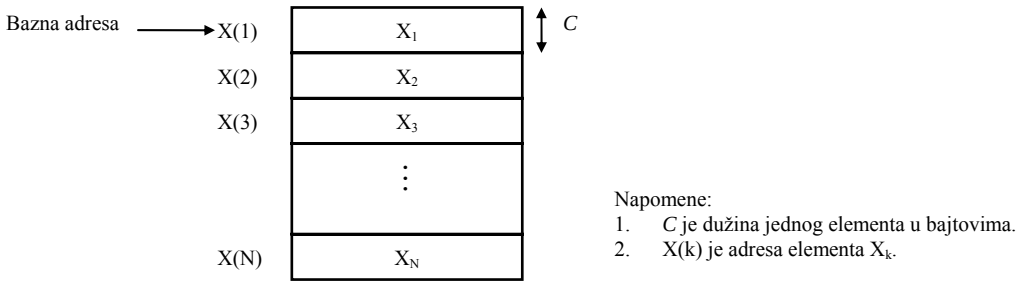
U Tabeli 3.1 prikazano je izračunavanje adresa za različita polja, a na slici 3.1 njihova organizacija u memoriji.

Tab. 3.1. Adresiranje jednodimenzionalnih polja.

Veličina elementa tabele	C	j-ta lokacija	Dužina polja (u bajtovima)
Bajt	1	X(j) = X(1) + (j-1)	N
Reč	2	X(j) = X(1) + 2*(j-1)	$2N$
Duga reč	3	X(j) = X(1) + 4*(j-1)	$4N$
Nizovi (fiksne dužine)	4	X(j) = X(1) + k*(j-1)	$k \times N$

Napomene:

1. N je broj elemenata dužine C bajtova.
2. Opseg indeksa je $1 \le j \le N$. Prva adresa je X(1) i sadrži vrednost X_1.



Sl. 3.1. Smeštanje polja u memoriji.

Kod mikroprocesora MC68020 se za adresiranje elemenata polja koriste adresni načini rada prikazani u Tabeli 3.2.

Tab. 3.2. Adresiranje polja kod MC68020.

Adresni način rada mikroprocesora MC68020	Tipična upotreba
Predekrementni	Za adresiranje elemenata dužine bajt, reč ili duga reč u opadajućem redosledu
Postinkrementni	Za adresiranje elemenata dužine bajt, reč ili duga reč u rastućem redosledu
(PC) relativno sa razmeštajem	Za lociranje elementa na fiksnoj poziciji u odnosu na baznu adresu
ili	
(An) sa razmeštajem	
(PC) relativno sa indeksiranjem	Za lociranje elementa na proizvoljnoj poziciji korišćenjem indeksnog registra
ili	
(An) sa indeksiranjem	
(PC) relativno sa umnoženim indeksom	Za lociranje bajta (faktor umnožavanja 1), reči (faktor umnožavanja 2), duge reči (faktor umnožavanja 4) ili četvorostruke reči (faktor umnožavanja 8) u polju; element se locira indeksnim adresiranjem
ili	
(An) sa umnoženim indeksom	
(PC) relativno sa baznim razmeštajem i indeksiranjem	Za lociranje elementa polja definisanog baznim razmeštajem i baznom adresom koja je u (PC) ili (An)
ili	
(An) sa baznim razmeštajem ili indeksiranjem	

Primer 3.1:

Instrukcijom

```
MOVE.W    (A1)+,D1
```

vrši se prenos 16-bitne vrednosti adresirane od strane A1 u (D1)[15:0]. A1 se zatim inkrementira za 2 i pokazuje na sledeću vrednost u polju.

Neka su data dva polja čiji su elementi dužine tipa reč. Ako (A1) pokazuje na prvi elemenat X(1), a (A2) pokazuje na Y(1), tada sekvencom instrukcija

```
MOVE.W    (4,A1),D1    ; (X(3))
MOVE.W    (4,A2),D2    ; (Y(3))
CMP.W     D1,D2        ; kompariranje (Y(3)-(X(3))
```

se kompariraju Y3 i X3 locirani na adresama Y(3) i X(3), respektivno.

Kod instrukcije

```
MOVE.W    (0,A1,D1.W),D1
```

bazna adresa koja se čuva u A1 i LS 16-bitna vrednost u D1 čine indeks.

Instrukcijom

```
MOVE.W    (A1, D1.W*4),D2
```

sa (D1)[W]=*n* adrese izvornog operanda na lokaciji

(A1)+4**n*

koja je *n*-ta duga reč u polju kada je *n* = 0, 1, 2, ... definiše indeks polja.

Instrukcijom

```
MOVE.B    (4,A1,D1.W*8),D2
```

adresiraće se prvi elemenat polja izvornog operanda na adresi (A1)+4 a preskočiće se prva četiri bajta informacije zaglavlja na početku polja čija je bazna adresa definisana sa (A1).

3.1.1. Instrukcija CMP2

CMP2 (Compare Register Against Bounds) se koristi da odredi da li je registarska vrednost u okviru numerički definisane vrednosti za donju i gornju granicu

$$\text{Donja granica} \le R(n) \le \text{Gornja granica}$$

gde Rn može biti An ili Dn.

Sintaksa naredbe CMP2 prikazana je u Tabeli 3.3a. Kao rezultat izvršenja instrukcije CMP2 postavljaju se markeri uslova C i Z. Posle CMP2 instrukcije obično sledi instrukcija Bcc u obliki BEQ, BNE, BCC, BCS ili BHI. Uslovi grananja su definisani u Tabeli 3.3b.

Tab. 3.3. CMP2 instrukcija.

(a) Sintaksa instrukcije.

Sintaksa	Adresni načini rada	Efekat

CMP2.<l> <EA>,<Rn>	Izvor: upravljački načini rada
	Registar: bilo koji An ili Dn
		Markeri:
		IF donja granica < (Rn) < gornja granica
		THEN
		C={0}, Z={0}
		ELSE
		IF (Rn) = donja granica or (Rn)= gornja granica
		THEN
		Z={1}, C={0}
		OTHERWISE
		C={1}, Z={0}

Napomene:
1. <l> = B, W ili L.
2. Upravljački načini rada obuhvataju sve načine za obraćanje memoriji osim - (An) ili (An)+.
3. U memoriji:
   <EA>←DONJA GRANICA
   <EA>←GORNJA GRANICA
   (bajt), 2 (reč) ili 4 (duga reč)
4. Donja granica mora biti algebarski manja (ili jednaka) od gornje granice.
5. Ako je <An> registar koji se ispituje, operandi obima bajt ili reč se znakovno proširuju do 32 bita ovom instrukcijom.

(b) Uslovi grananja posle CMP2 instrukcije.

Instrukcija	Kodovi uslova	Uslov
BEQ	Z={1}	(Rn) jednakj jednoj od granica
BNE	Z={0}	(Rn) nije jednak nijednoj od granica
BCC	C={0}	(Rn) unutar granica
BCS	C={1}	(Rn) izvan granica
BHI	C={0} i Z={0}	(Rn) je unutar granica ali nije jednak nijednoj od granica

Primer 3.2:

Instrukcijom

CMP2.L $20000,D2

komparira se 32-bitna vrednost koja se čuva u D2 sa 32-bitnom vrednošću koja se čuva počev od $20000. Poređenje je tipa

$$(\$20\,000) \le (D2).L \le (\$20\,004)$$

gde se u ($20000) čuva donja granica a u ($20004) gornja granica. Opseg svake granice kod neoznačenih celobrojnih vrednosti je $00000000 do $7FFFFFFF.

3.2. Dvodimenzionalna polja

Generalizacija jednodimenzionalnog polja je višedimenzionalno polje elemenata. Elementi višedimenzionalnog polja se specificiraju sa više indeksa. U Tabeli 3.4 prikazano je dvodimenzionalno $M*N$ polje gde je M broj vrsta, a N broj kolona. Elemenat i-te vrste i j-te kolone se označava kao X_{ij} gde je $1 \le i \le M$ i $1 \le j \le N$.

Tab. 3.4. Višedimenziona polja.

(a) Opšti oblik

$$\begin{matrix} X_{11} & X_{12} & \cdots & X_{1N} \\ X_{21} & X_{22} & \cdots & X_{2N} \\ . & . & & . \\ . & . & & . \\ . & . & & . \\ X_{M1} & X_{M2} & \cdots & X_{MN} \end{matrix}$$

(b) 3×3 primer

$$\begin{matrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ X_{21} & X_{22} & X_{33} \end{matrix}$$

(c) Smeštanje po kolonama polja dimenzija 3×3 počinje od lokacije $1000

Adresa (heksadecimalno)		Element polja
X(1,1)	$1000	X_{11}
X(2,1)	$1002	X_{21}
X(3,1)	$1004	X_{31}
X(1,2)	$1006	X_{12}
X(2,2)	$1008	X_{22}
X(2,3)	$100A	X_{32}
X(1,3)	$100C	X_{13}
X(2,3)	$100E	X_{23}
X(3,3)	$1010	X_{331}

Jedan veoma važan detalj koga treba uočiti u ovom slučaju odnosi se na način organizacije podataka dvodimenzionalnog polja u memoriji.

Ako se polje X dimenzija $M*N$ memoriše sekvencijalno po vrstama počev od adrese X(1,1) na sledeći način

X(1,1), X(1,2), ... , X(1,N), X(2,1), ... , X(2,N), ... , X(M,N)

za memorisanje kažemo da je tipa "row-major".

Alternativna šema je "column-major", kod koje se sukcesivni elementi adresiraju kao

X(1,1), X(2,1), ... , X(M,1), X(1,2), ... , X(M,2), ... , X(M,N)

"Column-major" oblik se koristi kod FORTRANA-a.

Nakon što je način memorisanja izabran, indeksi specificiranog elementa se mogu odrediti na nekoliko načina. Kod dvodimezionalnog $M*N$ polja adresa se određuje kao

$$X(i,j)=\text{bazna adresa}+C_1*(j - 1)+C_2*(i - 1)$$

pri čemu je prva adresa na lokaciji X(1,1), a C_1 i C_2 su konstante.

U Tabeli 3.5 prikazan je način određivanja adrese elementa dvodimenzionalnog polja

$$X(i,j)=B0+C*[(i - 1)+M*(j - 1)]$$

gde je $1 \le i \le M$ i $1 \le j \le N$, B0 je bazna adresa a C dužina elementa u bajtovima.

Tab. 3.5. Adresiranje višedimenzionalnih polja.

Smeštanje polja	Adresa od X(i,j)
Smeštanje po kolonama X(1,1), X(2,1), ...	$B_0 + C*[(i - 1) + M \times (j - 1)]$
Smeštanje po vrstama X(1,1), X(1,2), ...	$B_0 + C*[(j - 1) + N \times (i - 1)]$

Napomene:

1. B_o je bazna adresa polja $X(i,j)$ čiji su elementi dužine C bajtova.
2. Opseg indeksa je sledeći:
   vrste: $1 \le i \le M$
   kolone: $1 \le j \le N$

3.3. Povezane liste

Kod manipulacije sa poljima, adresa narednog elementa kome se pristupa se određuje dodavanjem konstante adresi tekućem elementu. Na primer, kod jednodimenzionalnog polja

$$X(j+1)=X(j)+C$$

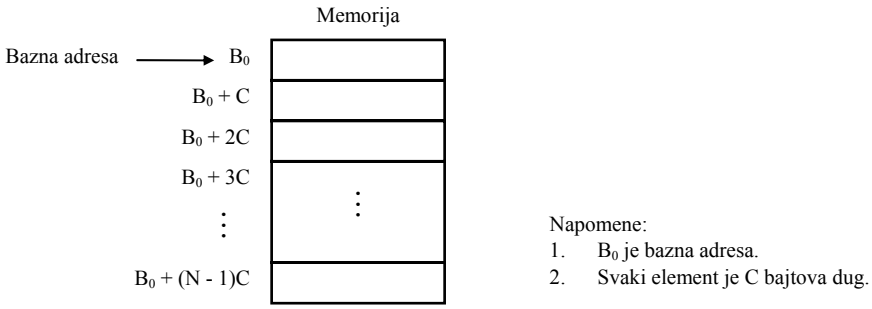
gde je C dužina elementa u bajtovima.

Elementi su uređeni sukcesivno i, kao što je prikazano na slici 3.2, smešteni su u kontinualnim memorijskim blokovima.

Nasuprot poljima, povezana lista je struktura podataka koja ne zahteva kontinualno memorisanje elemenata. Organizacija povezane liste od pet stavki prikazana je na slici 3.3.

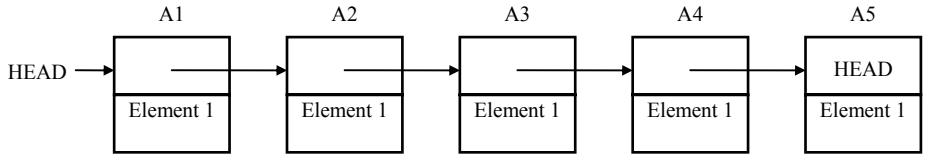
Svaki elemenat u listi sadrži:

a) pokazivač (adresu) ili vezu (link) ka sledećem elementu u listi,
b) podatke (data item).

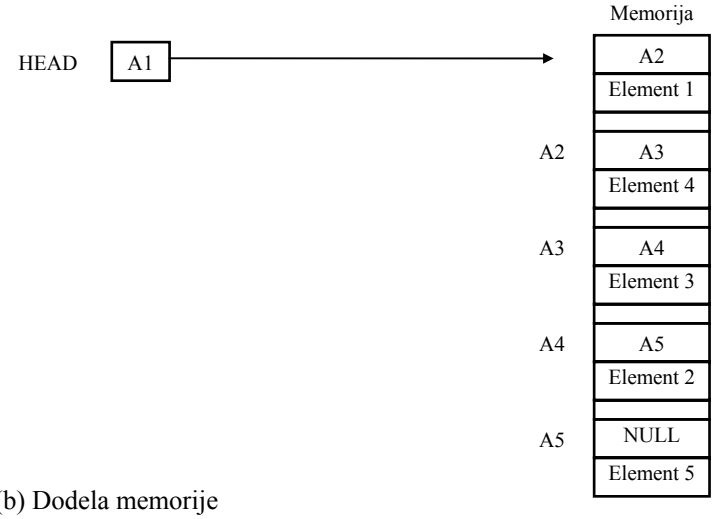


Sl. 3.2. Sekvencijalno smeštanje polja.

Lista prikazan na slici 3.3 ima jednosmernu vezu jer je moguće odrediti samo narednu stavku. Pokazivač liste je smešten na adresi HEAD. Zadnji elemenat, stavka 5 na slici 3.3, sadrži specijalni simbol nazvan NULL koji ukazuje na kraj liste. kada lokacija HEAD sadrži vrednost NULL lista je prazna. Na slici 3.3b prikazan je jedan od mogućih načina smeštanja stavki (elemenata) liste u memoriji.



(a) Prosto povezana lista



(b) Dodela memorije

Sl. 3.3. Alokacija za neuređenu povezanu listu.

3.3.1. Korišćenje CAS i CAS2 instrukcije kod povezanih listi

Kod MC68020 koriste se dve instrukcije za manipulaciju sa povezanim listama. Instrukcija CAS (Compare and Swap) se može koristiti za ubacivanje ili izbacivanje elemenata u jednoj povezanoj listi. Ona se koristi za posebne programske prednosti, ali se njeno izvršenje zasniva na korišćenju nedeljivog (read-modify-write) memorijskog ciklusa za pristup, pomoću koga se ažurira HEAD liste. CAS2 instrukcija se može koristiti za ažuriranje dvostruko povezanih listi, a zasniva se na korišćenju nedeljivog memorijskog ciklusa za pristup. Više detalja o ovim instrukcijama biće dato kasnije.

3.4. Korišćenje potprograma i prenos parametara

Korišćenje potprograma ili procedura je važna programska tehnika da se kreiraju modularni programi kod kojih svaki potprogram obavlja specifični zadatak u okviru potprograma. Kod MC68020 poziv potprograma se obavlja instrukcijom

JSR <SUBR>

koja prvo uslovljava da se adresa povratka u program iz koga se vrši poziv prvo sačuva u magacinu. Zatim se upravljanje prenosi na adresu potprograma <SUBR>. Na osnovu ovoga evidentno je da se prenos upravljanja kod MC68020 obavlja veoma jednostavno.

Sa druge strane, kada se vrši prenos podataka između pozivnog programa i potrpograma dostupan je veći broj metoda za prenos informacije.

Informacija koja je potrebna potprogramu definiše se u zavisnosti od parametara koji omogućavaju da potprogram manipuliše vrednostima. Svakim pozivom potprograma omogućava se da on manipuliše nad različitim vrednostima, koje zovemo argumenti, a predaju se kao parametri. Tipičan primer korišćenja potprograma u FORTRAN-u je prikazan na slici 3.4.

Potprogram SUBR ima parametre A, B i C. On se može pozivati sa različitim argumentima pod uslovom da su argumenti isti tipovi podataka (FP, celobrojne vrednosti i dr.), kao i parametri koji su definisani u toku deklaracije potprograma. Simbolička imena argumenata u datom primeru zamenjuju se od strane kompilatora kao adrese.

Mehanizam definisanja parametara i prenos argumenata potprograma kod asemblerskog jezika je dosta složeniji u odnosu na HLL. Za čuvanje argumenata koriste se registri procesora, sistemski magacin, ili fiksne memorijske lokacije. Takođe, za kreiranje okvira magacina se koriste instrukcije LINK i UNLINK. Okvir je memorijski blok rezervisan u magacinu koji se koristi za čuvanje adrese povratka, argumenata i lokalnih promenljivih, ako postoje.

Pozivajući program	Potprogram
.	SUBROUTINE SUBR(A,B.C)
.	.
.	.
CALL SUBR(X,Y,Z)	.
.	RETURN
.	END
CALL SUBR(1.0,3.0,RESULT)	
.	
.	
.	
CALL SUBR(A(1),W,ANS)	
.	
.	
.	
END	

Sl. 3.4. Korišćenje potprograma kod FORTRANA.

3.4.1. Prenos argumenata potprogramu

Parametri koji definišu argumente za prenos između potprograma i programa iz koga se vrši poziv mogu biti vrednosti podataka, adrese ili kombinacija i jednog i drugog. Kada se predaje mali broj argumenata oni se

prenose direktno preko registara. Kada je broj promenljivih veliki ili kada se prenose strukture, predaju se adrese. Kao što je prikazano u Tabeli 3.6 postoji nekoliko načina za prenos vrednosti ili adresa između programa.

Tab. 3.6. Metodi prenosa argumenata.

Tip	Opis	Komentari
Registar	Pozivajuća rutina puni unapred definisane registre vrednostima ili adresama.	Broje parametara je ograničen Dinamički
Stek	Pozivajuća rutina smešta vrednosti ili adrese u stek.	Adresa povratka se mora zapamtiti tokom obrade i obnoviti pre povratka
Parametarska oblast	Definisane su memorijske oblasti koje sadrže vrednosti ili adrese.	Statički, ako su oblasti definisane tokom asembliranja. Dinamički, ako se bazna adresa oblasti prenosi preko registra.
In-line	Vrednoti ili adrese su zapamćene iza poziva. Potrogram računa lokaciju parametara.	Statički

Argumenti koji se predaju potprogramu se definišu kao ulazni parametri, a rezultati mogu biti vrednosti ili adrese i odgovaraju izlaznim parametrima potprograma. Kod MC68020 moguće je koristiti kombinacije ovih metoda.

Registarski prenos

Najjednostavniji prenos se ostvaruje preko skupa registara. Vrednosti podataka se prenose preko registara Dn, a adrese koje ukazuju na podatke ili početne adrese strukture podataka, preko registara An.

Prednost ovakvog metoda je veoma brz prenos, ali je broj parametara ograničen na 15. Projektanti potprograma i programa iz koga se vrši poziv moraju se usaglasiti koji će se registri koristiti za prenos argumenata.

Primer 3.3:

Sekvencom instrukcija

```
MOVE.W    VREDNOST,D1 ; Podatak
MOVEA.L   ADDTAB,A1   ; Pokazivač
LEA       HEAD,A2     ; adresa HEAD-a
JSR       SUBR
```

postavlja se 16-bitna vrednost D1, pokazivač adresa u lokaciji ADDTAB u A1, i adresa HEAD u A2. Potprogram SUBR može pristupati vrednostima u registrima direktno.

Prenos preko magacina

Magacin se koristi za prenos argumenata na taj način što se vrednosti ili adrese smeštaju u magacin pre nego što se obavi poziv potprograma. Kod MC68020 moguće je definisati privatni magacin koristeći An registre, koji će se koristiti kao pokazivači magacina. Za smeštanje vrednosti u magacin koriste se predekrementirajući ili postinkrementirajući načini adresiranja.

Primer 3.4:

Sekvencom instrukcija

```
LEA       MAGP,A1       ; adresa magacina u A1
MOVE.W    VRED1,(A1)+   ; smesti prvu vrednost
MOVE.W    VRED2,(A1)+   ; smesti drugu vrednost
JSR       SUBR
```

postavlja se A1 kao pokazivač magacina i predaju se dve vrednosti. Potprogram pristupa vrednostima na sledeći način

```
MOVE.W    -(A1),D2      ; druga vrednost
MOVE.W    -(A1),D1      ; prva vrednost
```

Primer 3.5:

Kada se sistemski magacin koristi za prenos argumenata, adresa povratka, u trenutku kada izvršenje potprograma počne, nalazi se na vrhu magacina. Ova vrednost se mora sačuvati pre ngo što potprogram izbavi argumente iz magacina. Kada izvršenje potprograma završi, adresa povratka mora se ponovo smestiti na vrh magacina pre nego što se izvrši instrukcija RTS.

Sekvenca poziva kod ovog metoda predaje parametara ima oblik

```
        PEA     ADDR        ; push adresu
        MOVE.W  VRED1,-(SP) ; push vrednost
        JSR     SUBR
```

U magacin se prvo smešta adresa ADDR, a zatim vrednost sa lokacije VRED1, i na kraju adresa povratka. Kako je (PC) na vrhu magacina, on se mora sačuvati, a zatim obnoviti pre nego što se obavi povratak. Sekvenca naredbi koja obavlja tu aktivnost ima oblik

```
        MOVE.L      (SP)+,A1    ; sačuvaj (PC) privremeno
        MOVE.W      (SP)+,D1    ; pribavi podatak
        MOVEA.L     (SP)+,A2    ; pribavi adresu
                ⋮               ; obrada
        MOVE.L      A1,-(SP)        ; obnovi (PC)
        RTS
```

Memorijske lokacije za argumente

Kada se prenosi veliki broj parametara, parametarska oblast mora biti postavljena u memoriju. Ova oblast sadrži, u unapred određenoj sekvenci, vrednosti ili adrese kojima se pristupa od strane potprograma, nakon što je preneta početna adresa oblasti.

Program iz kojeg se vrši poziv treba da postavi parametarsku oblast na sledeći način

```
        MOVE.W      VAL1,PRMAREA    ; smeštanje podataka
        MOVE.W      VAL2,PRMAREA+2  ; drugu reč
                ⋮
        MOVE.W      VAL5,PRMAREA+8  ; petu reč
        LEA         PRMAREA,A1      ; smeštanje adrese
        JSR         SUBR
                ⋮
PRMAREA DS.W        5               ; rezervisana oblast
        END
```

Pet reči se definišu kao parametri. Potprogram može da pristupa vrednostima koristeći indirektno adresiranje sa razmeštajem. Na primer, instrukcijom

```
        MOVE.W      (6,A1),D1
```

obavlja se prenos četvrte reč u D1.

Postoji veći broj mogućnosti za definisanje parametarske oblasti u memoriji.

In-line kodiranje

Drugi metod da se preda vrednost potprogramu je da se kodiraju vrednosti nakon naredbe poziva potprograma. Ovaj način se zove In-line kodiranje, a definiše vrednosti argumenata koje su konstante i ne menjaju se posle asembliranja. Ove vrednosti se mogu definisati kao DC direktive i slede posle poziva.

Primer 3.6:

Sekvencom instrukcija

```
        JSR     SUBR
        DC.W    1 ; In-line parametar
```

32-bita (PC) se smeštaju u sistemski magacin nakon poziva potprograma. Ova adresa pokazuje na lokaciju argumenata u nizu naredbi.

Sekvencom instrukcija koja se izvršava od strane potprograma puni se argument u LS deo D1 i pokazuje se na adresu povratka.

```
        MOVEA.L     (A7),A0 ; pribavlja se vrednost PC-a
        MOVEA.W     (A0)+,D1        ; pribavlja se argument, inkrementira
        MOVE.L      A0,(A7) ; push nova adresa povratka
                ⋮
        RTS
```

Prvom naredbom vrednost (PC)-a iz magacina se smešta u A0. A0 zatim adresira argument i inkrementira se za +2, argument se smešta u D1. Narednom instrukcijom smešta se u magacin korektna adresa povratka.

Okviri magacina

Jedna od najvažnijih stavki kod projektovanja potprograma uključuje koncept transparentnosti. Prostije rečeno, kada potprogram završi sa izvršenjem on ne treba da ima vidljive efekte, sa izuzetkom na to kako je on povezan sa programom iz koga je izvršen poziv. Na primer, potprogram ne treba da menja vrednost u bilo kom registru, sa izuzetkom ako se rezultat predaje preko registra. U prethodnim primerima za prenos parametara i povratne adrese bio je korišćen magacin.

Koncept korišćenja magacina za memorisanje podataka koji se privremeno koriste u toku izvršenja potprograma se može proširiti definisanjem okvira magacina.

Okvir magacina je memorijski blok koji je deo magacina, a koristi se za čuvanje adrese povratka, ulaznih parametara, izlaznih parametara i lokalnih promenljivih. Lokalne promenljive su one vrednosti koje se koriste u toku izvršenja potprograma, a koje se ne prenose programu iz kojeg je izvršen poziv.

Korišćenje okvira magacina kod MC68020

Kod multiprogramskih sistema, nekoliko nezavisnih zadataka mogu koristiti isti potprogram. Na primer CRT terminali mogu se povezati na sistem ali tako da dele istu U/I rutinu. Kako operativni sistem ima ulogu komutatora između terminala, moguće je da se U/I rutina jednog prekine, a upravljanje se privremeno preda drugom terminalu. Svi podaci koji su pridruženi prvom terminalu, a koriste se od strane U/I rutine, moraju se sačuvati tako da kada prvi terminal dobije ponovo pravo upravljanja, U/I rutina nastavi sa onog mesta gde je bila prekinuta. Ovakav način korišćenja zahteva "reentrant" rutine kod kojih se vrednosti podataka u oblasti koja pripada programskoj memoriji sami po sebi ne menjaju u toku izvršenja. Bilo koja vrednost koja se menja smešta se u magacin. Na ovaj način program je u potpunosti izdvojen od podataka nad kojima on manipuliše. Specijalan zadatak je rekurzivna rutina, koja zove samu sebe i zbog toga je "self-reentrant". Okviri magacina omogućavaju da se "reentrant" i rekurzivne rutine lako kreiraju.

Okvir magacinma kod MC68020 se kreira od strane programa iz koga se vrši poziv, a potprogram koristi LINK i UNLK instrukcije. Sintaksa i princip rada ovih instrukcija je definisan u Tabeli 3.7. Pristup promenljivim u magacinu od strane potprograma se ostvaruje korišćenjem ofseta (ili indeksiranjem) u odnosu na bazni registar koji se zove pokazivač okvira. Pokazivači okvira se ne menjaju u toku izvršenja potprograma.

Tab. 3.7. Efekat instrukcija LINK i UNLK.

Instrukcija	Sintaksa		Efekat
Link	LINK.<l_1>	<An>,#<disp>	1. (SP)←(SP) - 4; ((SP))←(An)
			2. (An)←(SP)
			3. (SP)←(SP) + <disp>
Unlink	UNLK	<An>	1. (SP)←(An)
			2. (An)←((SP)); (SP)←(SP) + 4

Napomene:

1. <disp> je 32-bitni ili 16-bitni znakovno prošireni ceo broj. Negativni razmeštaj se specificira za dodelu memorijske oblasti za stek.
2. <l_1> = W ili L.