Poglavje 5

IZJEME

V poglavju 2.2.4 smo našteli stanja, v katerih se lahko nahaja mikroprocesor: *normalno*, *ustavljen* ali *strežba izjem*. V tem poglavju se bomo izrecno posvetili zadnjemu stanju.

5.1 Splošni pojmi

Izjema je *dogodek* (signal, napaka ali ukaz), ki povzroči, da se normalni način delovanja – izvajanja ukazov iz programa – prekine in izvede *strežni program izjeme*, ki ga je programer predvidel kot reakcijo nanjo. Ko se ta program zaključi, se glavni potek izvajanja nadaljuje tam, kjer je bil prekinjen; običajno zahtevamo, da prekinjeni program (razen časovnega zamika) ne čuti, da se je vmes izvajala strežba izjeme.

Viri za izjeme so lahko *znotraj* ali *zunaj* mikroprocesorja. Notranji viri izjem so na primer *ukazi* (TRAP, TRAPV, CHK, DIV..), posebni načini delovanja (npr. sledenje[1]) ali *napake* (napake v naslavljanju, nedovoljeni ukazi, prekoračitev privilegijev ipd.) Zunanji viri izjem pa so prekinitve, signali za razne napake, RESET ipd.

Razlikujemo dve kategoriji izjem:

- *pasti:* pasti so običajno notranje generirane izjeme, ki prestrezajo posebne dogodke v delovanju sistema. Ti so lahko nepričakovani in nezaželeni, kot so npr. deljenje z 0 ali napaka v naslovu; sem spada tudi napaka na vodilu, ki je sicer zunanja izjema. Med pasti prištevamo še sledenje, pri katerem se delovanje programa prestreže po vsakem ukazu.

- *prekinitve:* običajno jih generirajo periferne naprave, ko želijo začasno prekiniti potek delovanja mikroprocesorja in izvesti določeno strežno rutino. Po zaključku se nadaljuje obdelava prekinjenega programa, po možnosti s čim manj posledicami prekinitve. Ta postopek se imenuje *zamenjava konteksta*[2] (vsebine). Uporaba preki-

[1] Trace mode
[2] Context switch

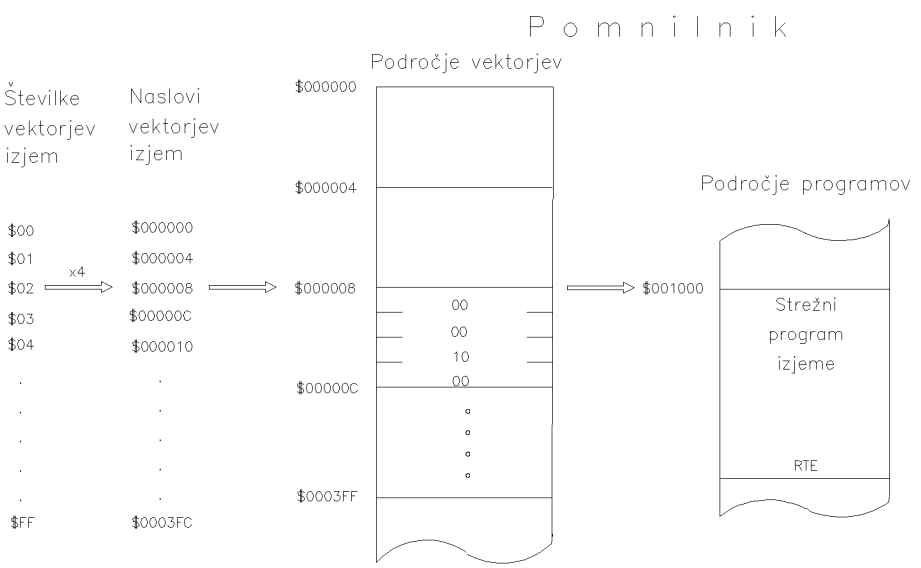
nitev je dodatno obdelana v poglavju o sinhronizaciji prenosa podatkov s perifernih naprav 7.3.2.

Zgoraj omenjeni *strežni program izjeme* je program, ki se naj izvede ob pojavu izjeme. Običajno je *rezidenčno* (stalno) naložen v pomnilniku in se vedno požene v nadzornem načinu. Programer pripravi program in njegov začetni naslov vpiše v t.i. *vektor izjeme*; to je posebna sistemsko definirana lokacija v pomnilniku, katere naslov običajno določi proizvajalec mikroprocesorja. Pri nekaterih mikroprocesorjih lahko uporabnik s posebnim (t.i. *baznim*) registrom področje vektorjev prestavlja na poljubno mesto.

Ko se zgodi izjema, mikroprocesor poišče njen vir in iz pripadajočega vektorja prenese naslov strežne rutine v PC, s čimer se izvede skok vanjo. Na pomnilniškem področju, kjer so vektorji izjem, je v končnih aplikacijah običajno bralni pomnilnik (ROM), v katerega fiksno sprogramiramo naslove strežnih rutin.

Priporočljivo je, da vsebine vektorjev izjem vedno določimo, tudi za izjeme, ki jih ne pričakujemo; če spregledamo možnost in se izjema vendarle zgodi, njen vektor pa ni določen, nadaljnje izvajanje ne bo več definirano. Za takšne izjeme običajno združimo strežne rutine v skupni diagnostični rutini. *Dolžina* vektorjev je določena z velikostjo naslovov, pri sodobnih mikroprocesorjih običajno 32 bitov oz. 4 zlogi.

Vektorje običajno podajamo po njihovih *številkah*, kakor je prikazano v primeru v nadaljevanju. Pri *MC68000* se področje vektorjev začne na naslovu 000000, od koder si vektorji zaporedoma sledijo. Ker so naslovi dolgi po 4 zloge, se naslov posameznega vektorja izračuna tako, da se njegova številka pomnoži s 4. To se v dvojiški aritmetiki izvede preprosto z dvema premikoma v levo. Primer je podan na sliki 5.1: številka izjeme "napaka na vodilu" je 2, naslov ustreznega vektorja je 8, kar je fiksno definirano pri *MC68000*. Programer je za začetek strežne rutine izbral naslov $001000 in ga vpisal v vektor (lokacijo v pomnilniku z naslovom $000008). V tem programu je predvidel, kaj se naj zgodi ob napaki na vodilu.



Slika 5.1: Določanje naslova strežne rutine za izjemo "napaka na vodilu" pri *MC68000*

V mikroračunalnikih, ki so namenjeni razvoju in ne končnim aplikacijam, je na teh naslovih pogosto RAM in naslove v vektorje (razen vektorja za inicializacijsko rutino ob resetu) vpišemo z inicializacijskim programom; s tem omogočimo, da jih lahko med delovanjem spreminjamo.

Zelo ugodno je, če tudi v sistemih, ki imajo na področju vektorjev ROM, skok na strežne podprograme – razen inicializacijskega – izvedemo posredno preko vsaj enega ukaza v RAMu: vektor za izjemo kaže na naslov v RAM-u, na katerem se nahaja brezpogojni skok na dejansko strežno rutino v ROM-u. Te brezpogojne skoke vpišemo v pomnilnik ob inicializaciji in jih lahko kasneje spremenimo tako, da kažejo na druge rutine. Tako dobimo *dinamične* vektorje. Ob inicializaciji v operacijskem sistemu se v te vektorje vpišejo naslovi sistemskih strežnih oziroma diagnostičnih rutin. Med delovanjem programa jih lahko poljubno spremenimo in s tem dosežemo, da se ob izjemi izvrši alternativna strežna rutina.

Stanje registrov v mikroprocesorju predstavlja trenutno okolje ali kontekst, v katerem deluje program. Koncept delovanja izjem zahteva, da zagotovimo nadaljevanje prekinjenega programa z istim kontekstom po zaključku strežbe izjeme, torej obdelave pripadajočega strežnega programa. Ker v mikroprocesorjih običajno obstaja le en nabor registrov, moramo vsebino tistih, ki jih bomo v strežnem programu spreminjali, ob prihodu izjeme shraniti v pomnilnik in jo ponovno naložiti po koncu strežbe. Nekateri mikroprocesorji, predvsem starejši z manjšim številom registrov, npr. *M6800*, so avtomatično ohranili vse. Pri mikroprocesorjih z velikim številom registrov to ne bi bilo smotrno, saj običajno ni potrebno ohraniti vseh, temveč le tiste, ki jih strežna rutina spremeni. Za to so nam na voljo posebni ukazi, npr. pri *MC68000* MOVEM, kjer z enim ukazom damo na sklad vse tiste registre, ki jih navedemo kot operande.

5.2 Izjeme pri MC68000

Za podrobnejši prikaz vrst in delovanja izjem bomo kot vzorčni primer spet uporabili *MC68000*. Opisane bodo izjeme, ki jih ta procesor prepozna in ki se pogosto pojavljajo tudi v drugih mikroprocesorjih. Izjeme bodo naštete po zaporednih številkah vektorjev. V nadaljevanju bo opisana njihova strežba.

5.2.1 Seznam izjem in njihovih vektorjev

0 *Reset: začetni SSP;* iz tega naslova se ob inicializaciji naloži nadzorni kazalec sklada. To je potrebno zato, da je kazalec in z njim področje sklada definirano od vsega začetka izvajanja inicializacijske rutine;

1 *Reset: začetni PC;* vektor, iz katerega se ob inicializaciji naloži PC - kazalec na inicializacijsko proceduro;

2 *Napaka na vodilu;* procesor naloži ta vektor, če je $\overline{BERR}$ signal na nizkem nivoju. To se lahko zgodi iz različnih vzrokov, najbolj tipični so: ni potrditve podatkov s periferne enote v predvidenem času (generira posebni timer); nedovoljen dostop

do podatkov (generira enota za upravljanje s pomnilnikom); med prenosom so bile ugotovljene napake itd..

3 *Napaka v naslovu;* MC68000 lahko naslavlja v pomnilniku dvo– in štirizložne besede le na sodih naslovih, zato nima naslovne linije A_0 (posamezne zloge dosega tako, da izbere zgornjo – s sodim – ali spodnjo – z lihim naslovom – polovico besede). Ob poskusu čitanja besede ali dolge besede z neparnega se zgodi napaka v naslovu;

4 *Nedovoljen ukaz;* prevzeti ukaz ni veljaven strojni ukaz *MC68000*; ta izjema se lahko uporabi tudi npr. pri testiranju programov (*break-point:* ob namenoma vnešenem napačnem ukazu se delovanje prekine prekine in izvrši strežna rutina predvidena za napačni ukaz. V njej omogočimo pregled registrov, pomnilnika, itd);

5 *Deljenje z nič;* pri ukazu DIV ali DIVU je bil divizor enak 0;

6 *CHK ukaz;* izjema se zgodi pri ukazu CHK, ki ima obliko
CHK ⟨ea⟩ D_n; ukaz primerja vsebino podatkovnega registra D_n z 0 in z vsebino na dejanskem naslovu ⟨ea⟩ in če je izven teh meja, sproži izjemo.

7 *TRAPV ukaz;* ukaz TRAPV testira bit V in sproži izjemo, če je postavljen, torej če se je v prejšnji operaciji zgodil preliv *(overflow)*;

8 *Prekoračitev privilegijev;* v programu, ki teče v uporabniškem načinu, smo poskušali uporabiti priviligiran ukaz, namenjen le nadzornemu načinu;

9 *Sledenje;* ob težavah pri iskanju napak lahko postavimo bit T v CCR na ena; po vsakem ukazu se nam bo odslej zgodila izjema, v katere strežnem programu omogočamo pregled stanja mikroprocesorja;

10 *Emulator ukaza 1010;* sorodna izjemi "nepravilni ukaz". Ta in naslednja izjema se zgodita, kadar se koda strojnega ukaza začne z $A oziroma $F. Ti ukazi pri *MC68000* niso implementirani, uporabnik pa ima možnost, da s pomočjo strežnih rutin emulira poljubni, pogosto potrebni ukaz. Ta se bo klical kot normalni strojni ukaz, njegova obdelava pa bo zahtevala nekaj več časa;

11 *Emulator ukaza 1111;* kot zgoraj. Princip emulacije ukazov je uporabljen v primeru ukazov za obdelavo realnih števil s plavajočo vejico: mikroprocesor MC68020 podpira priključitev aritmetičnega koprocesorja MC68881 s posebnimi ukazi, katerih operacijske kode (in tudi mnemoniki) se začenjajo z F. Iste programe lahko poganjamo tudi na mikroprocesorju MC68000 brez koprocesorja, če izdelamo ustrezno strežno rutino za emulator ukaza 1111. Ta mora prepoznati ukaz in izvesti pripadajočo rutino, ki izračuna rezultat aritmetične operacije.

12 - 14 rezervirano;

15 *neinicializirana prekinitev;* kadar pričakujemo, da bo periferija poslala ob prekinitvi vektor strežne rutine, ta pa ni bil predhodno vpisan (ob inicializaciji), bo poslan vektor "neinicializirana prekinitev";

16 - 23 rezervirano;

24 *lažna prekinitev;* v prekinitvenem prevzemnem ciklu se zgodi napaka na vodilu; najpogostejši vzrok za to je, če se po prekinitvi nihče ne odzove na zahtevo po prekinitvenem vektorju;

25 *Avtovektor - nivo 1;* te izjeme predstavljajo prekinitvene zahteve enot, ki niso sposobne generirati lastnih prekinitvenih vektorjev. Ob prekinitvi na določenem nivoju se izvede ustrezni avto-vektor. Takšen primer strežbe izjeme je opisan na koncu tega poglavja.

26 *Avtovektor - nivo 2*

27 *Avtovektor - nivo 3*

28 *Avtovektor - nivo 4*

29 *Avtovektor - nivo 5*

30 *Avtovektor - nivo 6*

31 *Avtovektor - nivo 7*

32 - 47 *Pasti* ali programske prekinitve; ob ukazu TRAP #no. se izvede ena od 16 izjem. Pasti se uporabljajo za klice sistemskih rutin in s tem za prehod iz uporabniškega v nadzorni način.

48 - 63 rezervirano

64 - 256 *vektorji za prekinitve;* teh 192 vektorjev je na razpolago za uporabo pri prekinitvah, ki jih prožijo enote, sposobne generirati vektorje. Njihove številke ob inicializaciji posredujemo perifernim napravam, ki kasneje ob prekinitvenem prevzemnem ciklu zahtevajo obdelavo strežne rutine, katere naslov je vpisan v vektorju z ustrezno številko.

5.2.2 Strežba izjem

Strežba izjem poteka v splošnem v naslednjih korakih:

- Ko procesor prepozna izjemo, najprej naredi interno kopijo statusnega registra, kot je bil pred tem. S tem omogoči, da ga lahko med strežbo izjeme poljubno spreminjamo.
- Postavi delovanje mikroprocesorja v nadzorni način in po potrebi izključi sledenje. V primeru prekinitev naloži prioritetni nivo v CR, da prepreči, da bi ista prekinitev prekinjala samo sebe.
- Prevzame številko vektorja; v primeru vektorskih prekinitev se izvede prekinitveni prevzemni cikel, v katerem procesor prečita številko vektorja iz enote, ki ga je prekinila. Številka vektorja se nato pomnoži s 4 (dvakrat premakne v levo).

- Shrani programski števec in kopijo statusnega registra na nadzorni sklad; s tem omogoči, da se bo po zaključku strežbe lahko nadaljeval prekinjen program;
- Naloži vsebino vektorja v PC in s tem sproži obdelavo strežne rutine za izjemo. Rutina se mora zaključiti z ukazom RTE, ki naloži s sklada kontekst programa, ki je bil prekinjen in ga tako nadaljuje.

Zaradi asinhronosti delovanja se lahko zgodi tudi več izjem v skoraj istem trenutku. Mikroprocesor jih prepoznava in razporeja njihovo strežbo po prioritetah, ki so razporejene v tri skupine, pri čemer je najvišja 0:

0 RESET, napaka na vodilu, napaka v naslovu: najbolj radikalne izjeme, prepoznajo in izvedejo se že na koncu urinega cikla;

1 prekinitve, sledenje, nepravilni in emulirani ukazi, prekoračitev privilegijev ipd.: prepoznajo se ob koncu ukaza ali cikla za prenos podatkov;

2 ukazi TRAP, TRAPV, CHK, deljenje z 0: prepoznajo se znotraj cikla ukaza.

Na sliki 5.2 je podrobneje prikazana strežba izjem pri *MC68000*. Opazujemo lahko, kako mikroprocesor reagira na notranje izjeme in oba načina prekinitev oziroma lažno prekinitev. Opazimo tudi, da je posebna pozornost posvečena napakam, ki se lahko zgodijo med strežbo izjeme in ki povzročijo novo izjemo ali dvojno napako, če se ponovijo.

Ko se zgodi izjema iz skupine 0 (razen RESET), se shrani na sklad več informacije kot v izjemah iz 1. in 2. skupine. Poleg PC in SR se shranijo še natančnejši podatki o dogajanju pred izjemo (glej sliko 5.3). Ker se delovanje procesorja ustavi znotraj ukaza, je pomembno, ali je bil že (pred)prevzet naslednji ukaz in se temu ustrezno poveča PC, predno se ga shrani na sklad.

5.2.3 Prekinitve

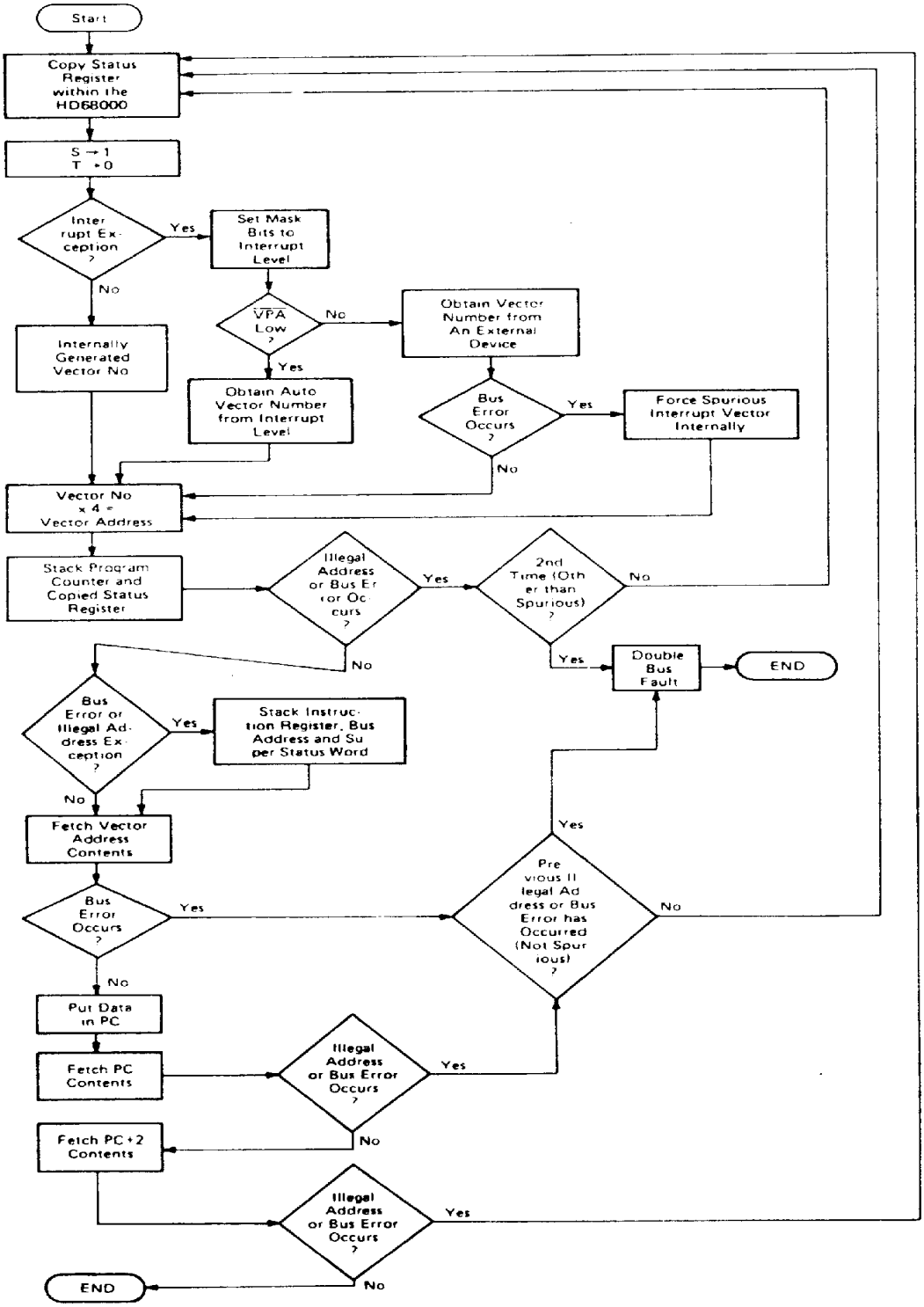
Prekinitve so izjeme, ki zaradi svoje pomembnosti zaslužijo posebno obravnavo. Proži jih neko predvideno dogajanje v okolju, ki ga opazi periferna naprava. Ta je priključena na mikroračunalniški sistem preko perifernega vmesnika. Poleg prenašanja podatkov je njena naloga tudi signalizirati mikroprocesorju, kadar je treba prekiniti trenutno obdelavo in izvesti neko akcijo. Za to ima na voljo prekinitvene linije, v primeru *MC68000* $\overline{IPL_{0..2}}$. S kombinacijo signalov na teh treh linijah lahko dosežemo sedem prekinitvenih nivojev (in stanje 111 brez prekinitev, glej tabelo 3.3 na strani 50). Ko mikroprocesor opazi nizek nivo na vsaj eni od teh linij, preveri, ali je prekinitveni nivo, ki ga podajajo le-te, višji od nivoja programa, ki se izvaja, in ki je vpisan v $I_{0..2}$ v statusnem registru. Če je višji, se bo obdelava prekinila in postregla se bo prispela prekinitev. Izjema so prekinitve na najvišjem, sedmem nivoju, ki se vedno postrežejo.

Strežba prekinitev se v tretjem koraku razlikuje od strežbe drugih izjem: po tem, ko skopira statusni register, vklopi nadzorni način in izklopi sledenje, mikroprocesor postavi

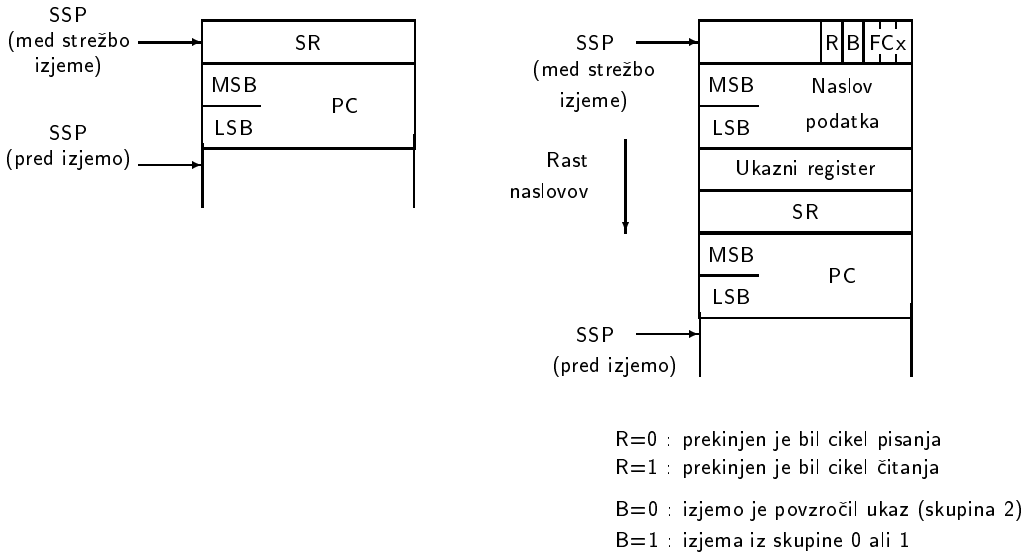
novo stanje bitov $I_{0..2}$ v statusnem registru, s čimer prepreči, da bi ga prekinjale manj pomembne prekinitve. Nato sproži postopek prevzemanja vektorja, ki je prikazan na sliki 5.4. To se dogaja v ciklu na vodilu, ki je podoben ciklu čitanja podatkov, in se od njega loči predvsem po drugačnih funkcijskih kodah (glej 3.2) in drugi funkciji naslovnega vodila; po njem se prenaša informacija o prekinitvenem nivoju. Ko mikroprocesor prevzame vektor, se nadaljuje običajna strežba izjeme.

Poseben primer nastopi, ko je prekinitev prožila naprava, ki ni sposobna generirati vektorja. Takšna naprava se na prekinitveni prevzemni cikel odzove z $\overline{VPA}$ namesto z vektorjem in $\overline{DTACK}$[3]. V tem primeru mikroprocesor privzame avtomatski vektor za prekinitve na ustreznem nivoju (glej prejšnje poglavje).

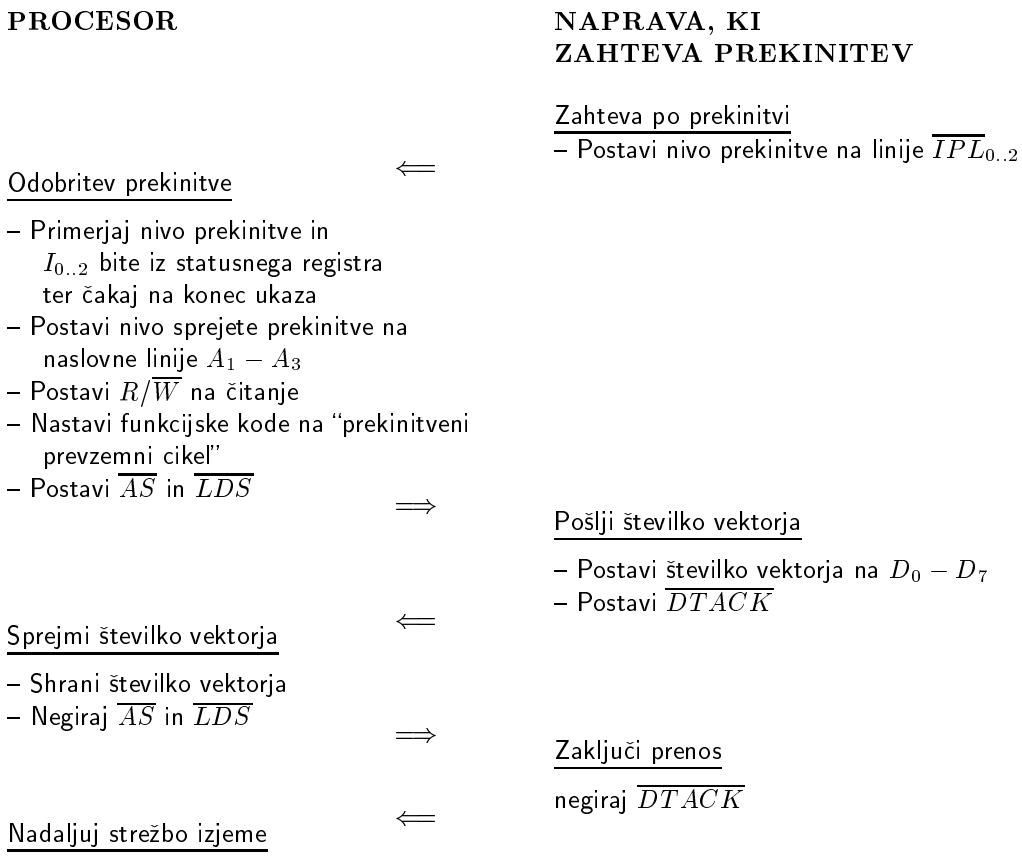
[3]Podoben primer smo srečali, kadar $MC68000$ komunicira s periferno napravo v semisinhronem načinu. Tudi tedaj se le-ta odzove z $\overline{VPA}$, mikroprocesor pa nato preide v drugi način delovanja, glej stran 49



Slika 5.2: Diagram poteka strežbe izjem pri *MC68000*



Slika 5.3: Slika sklada po izjemah pri MC68000



Slika 5.4: Prekinitveni prevzemni cikel