Navidezni pomnilnik (Virtualni pomnilnik)
(Podpora navideznemu pomnilniku na nivoju strojne opreme)

Kaj je navidezni pomnilnik

- Mehanizem, ki daje uporabniku vtis, da je glavnega pomnilnika več kot ga je v resnici.
  - Možno je izvrševanje programov, ki so obsežnejši od velikosti glavnega pomnilnika. (Neodvisnost velikosti programa od velikosti pomnilnika).
  - Možna je sočasna namestitev večjega števila programov, ki so v skupnem obsegu večji (resnične) velikosti pomnilnika. (Povečanje stopnje sočasnosti - večopravilnosti).

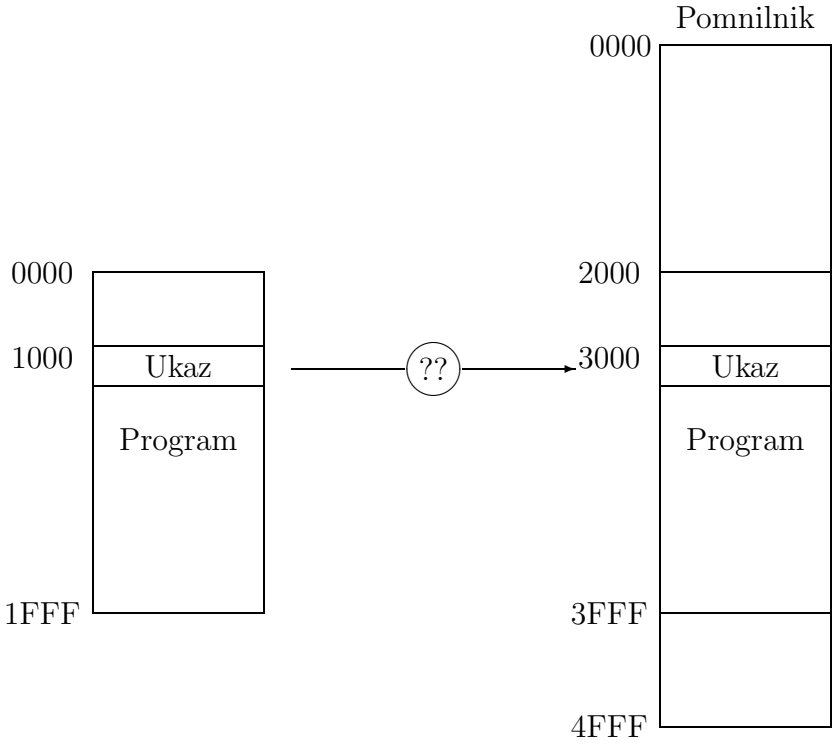
  To se doseže z nameščanjem v glavni pomnilnik samo tistih delov programa, ki so trenutno potrebni (na "zahtevo").
- Analize so pokazale, da za večino programov velja,
  - nekateri deli programov se le redko izvršijo,
  - vsi deli programa za izvršitev sploh niso potrebni (npr. nastavitve, diagnostika, pomoč...).
  - če že so, potem niso potrebni ves čas.

  Torej ni potrebe,m da bi se cel čas ali spoh nahajali v pomnilniku.
- Podlago za delovanje navideznega pomnilnika daje lastnost večine današnjih programov: lokalnost pomnilniških referenc.
- Koncept navideznega pomnilnika ni nov. Prvič je bil prikazan leta 1959 na University of Manchester v sistemu Atlas.
- V 60 letih je koncept navideznega pomnilnika prisoten na velikih računalnikih (IBM 360, CDC 7600).
- Množična prisotnost je razmeroma nova (osebni računalniki).

Koncept navideznega pomnilnika

Za razumevanje koncepta navideznega pomnilnika je bitvenega pomena, da ločimo logični naslov od fizičnega naslova.

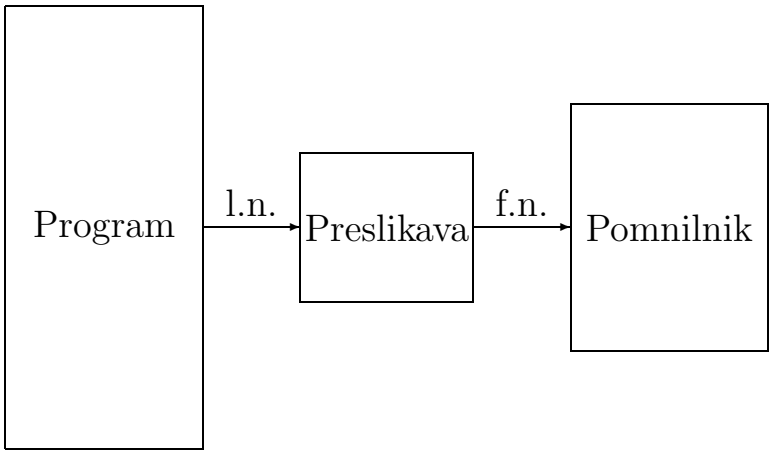
- Logični naslov je naslov znotraj programa, to je naslov določenega ukaza ali podatka. To je tudi naslov, ki ga generira procesor med izvrševanjem programa.
- Fizični naslov je naslov pomnilniške besede (glavnega) pomnilnika. To je torej naslovna kombinacija na naslovnem vidilu.
- Logični naslovi sestavljajo logično naslovno področje, to je naslovno področje programa. Velikost (obseg) logičnega naslovnega področja je dan z velikostjo programa.
- Fizični naslovi tvorijo fizično naslovno področje pomnilnika. Velikost (obseg) fizičnega naslovnega področja je dana z velikostjo pomnilnika.



- Programu dodelimo pomnilnik, npr. ukaz z logičnim naslovom 1000 namestimo v pomnilniško besedo s fizičnim naslovom 3000.
- Vprašanje: Kako in kdaj logične naslove preslikati v fizične naslove?

Preslikovanje logičnih v fizične naslove

- Statično (s prevajanjem, povezovanjem, nameščanjem) – pred izvršitvijo.
  - V času pisanja in prevajanja programa (s prevajanjem). Program pišemo ob predpostavki, da so logični naslovi enaki fizičnim (absolutni program). Program namestimo na dane (predvidene) fizične naslove in izvršimo.
  - V času povezovanja (s povezovanjem). Programe pišemo ob predpostavki, da bodo logični naslovi preslikani (preračunani) v ustrezne fizične naslove v času povezovanja ("linkanja") z drugimi programi oziroma deli programa. Posamezni deli programa se najprej prevedejo "relativno" na logični naslov 0000 in nato povežejo. Rezultat povezovanja je absolutni program. Program namestimo na dane (predvidene) fizične naslove in izvršimo.
  - V času nameščanja. Programu priredimo fizične naslove (preračunamo) na dane fizične naslove med nameščanjem v pomnilnik in ga tja tudi namestimo.
- Dinamično (med samim izvrševanjem). Ko procesor generira določen logični naslov, se le-ta s pomočjo ustreznega mehanizma (strojne opreme) preslika v ustrezen fizični naslov (Npr.: ko procesor generira naslov 1000 se le-ta preslika v prirejeni fizični naslov 3000). Na tem temelji izvedba navideznega pomnilnika.



Vprašanje: kako na učinkovit (hiter) način preslikovati logične naslove v fizične?

Upravljanje pomnilnika

- Obravnavanje navideznega pomnilnika sodi v širši kontekst upravljanja pomnilnika.
- Upravljanje pomnilnika je pomebna sestavina operacijskih sistemov.
- Strojna oprema mora dati ustrezno podlago za učinkovito izvedbo.
- Upravljanje pomnilnika je pomembno v vsakem (operacijskem) sistemu. V večopravilnem sistemu je upravljanje pomnilnika bistvenega pomena.
- Sistem za upravljanje pomnilnika je zadolžen za dodeljevanje, sproščanje in administriranje pomnilnika,
  - določa katerim programom/procesom dodeliti pomnilnik,
  - kdaj, kje, koliko in za koliko časa,
  - vodi pregled nad uporabo in preprečuje nepravilnosti.

Dodeljevanje pomnilnika

- Zvezno, program namestimo v "enem kosu", od najnižjega do najvišjega naslova.
  - Posamično zvezno. V pomnilnik namestimo samo en program (enoopravilnost).
  - Deljeno zvezno. V pomnilnik namestimo zvezno več programov (večopravilnost).
    - Pomnilnik je vnaprej deljen na področja danih in nespremenljivih velikosti ali "particije" (en program v eno particijo).
    - Pomnilnik ni vnaprej deljen na particije, velikost področij se spreminja in je odvisna od velikosti programov.
- Nezvezno, program razdelimo in namestimo v "več kosih". To je podlaga za izvedbo navideznega pomnilnika. Program delimo na:
  - enake dele ali strani (ostranjenje),
  - neenake dele ali segmente (segmentiranje),
  - kombinacijo obeh (segmentiranje z ostranjenjem).

Navidezni pomnilnik - povzetek

- izvajanje programov, ki niso v celoti nameščeni v pomnilnik, kar doseže
- z delitvijo programov na manjše dele (strani ali/in segmente),
- z nameščanjem v glavni pomnilnik samo tistih delov programa (strani, segmentov), ki so trenutno potrebni (na zahtevo). Preostanek programa ostane na zunajem pomnilnem mediju (disku),
- preslikavo logičnih naslovov v fizične med samim izvrševanjem programa (dinamično).

Bistvene prednosti:

- izvrševanje porgramov, ki so večji od fizičnega pomnilnika,
- velikost programov, ki jih lahko izvršimo, ni direktno odvisna od velikosti fizičnega pomnilnika,
- večja stopnja večopravilnosti,
- bolj učinkovito nameščanje (hitrejše, varčnejše).

Brez lokalnosti pomnilniških referenc bi koncept navideznega pomnilnika ne imel pravega smisla.

Opombe:

- Navidezno povečanje pomnilnika se da doseči tudi brez navideznega pomnilnika s prekrivki (angl. Overlay).
- Navidezni pomnilnik se da realizirati tudi brez ostranjenja ali segmentiranja z umikanjem programov na disk v celoti (angl. Swapping).
- Včasih zasledimo izraz navidezni pomnilnik za vsak sistem upravljanja pomnilnika z ostranjenjem ali segmentiranjem.

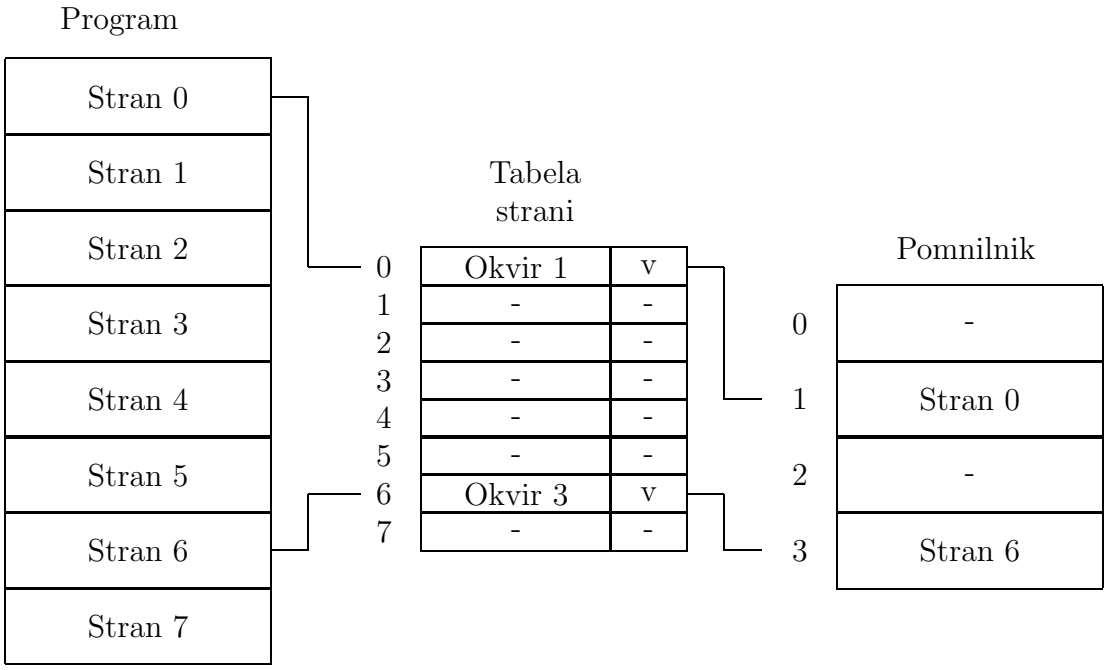
Možnosti za izvedbo navideznega pomnilnika

- Ostranjenje (angl. Demand Paging)
- Segmentiranje (angl. Demand Segmentation)
- Ostranjeno segmentiranje (angl. Demand Segmentation with Paging)

Ostranjen navidezni pomnilnik

- Logično naslovno področje programa delimo na enako dolge dele - **strani** (npr. 4 Kb).
- Fizično naslovno področje glavnega pomnilnika delimo na enako dolge dele - **okvirje strani** (Page Frames).
- Velikost strani je enaka velikosti okvirja strani.
- Katerokoli stran programa lahko namestimo v katerikoli okvir stani pomnilnika.
- Program ni zvezno nameščen v pomnilniku. Zvezno je nameščena samo posamezna stran.
- V sistemih z navideznim pomnilnikom so v pomnilnik nameščene samo nekatere (trenutno potrebne) strani.
- Logični naslovi (strani) se med izvajanjem programa preslikajo v fizične naslove (okvirjev strani) preko **tabele strani** (Page Table).
- V primeru "zadetka" (reference na stran, ki je v pomnilniku), se izvrševanje programa nadaljuje.
- V primeru "zgrešitve" oziroma "napake strani" (reference na stran, ki ni v pomnilniku), se izvrševanje programa prekine. Operacijski sistem namesti stran v pomnilnik, osveži vsebino tabele strani in obnovi izvrševanje programa.

Ostranjen navidezni pomnilnik - primer



Načelni potek preslikave

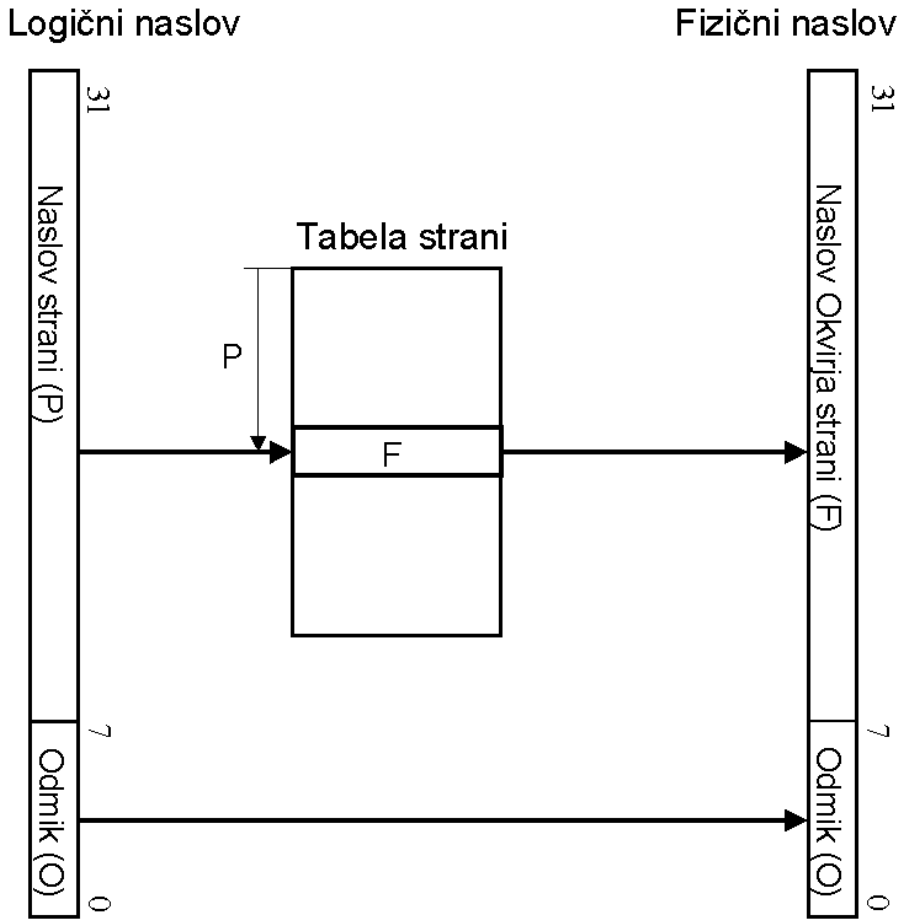
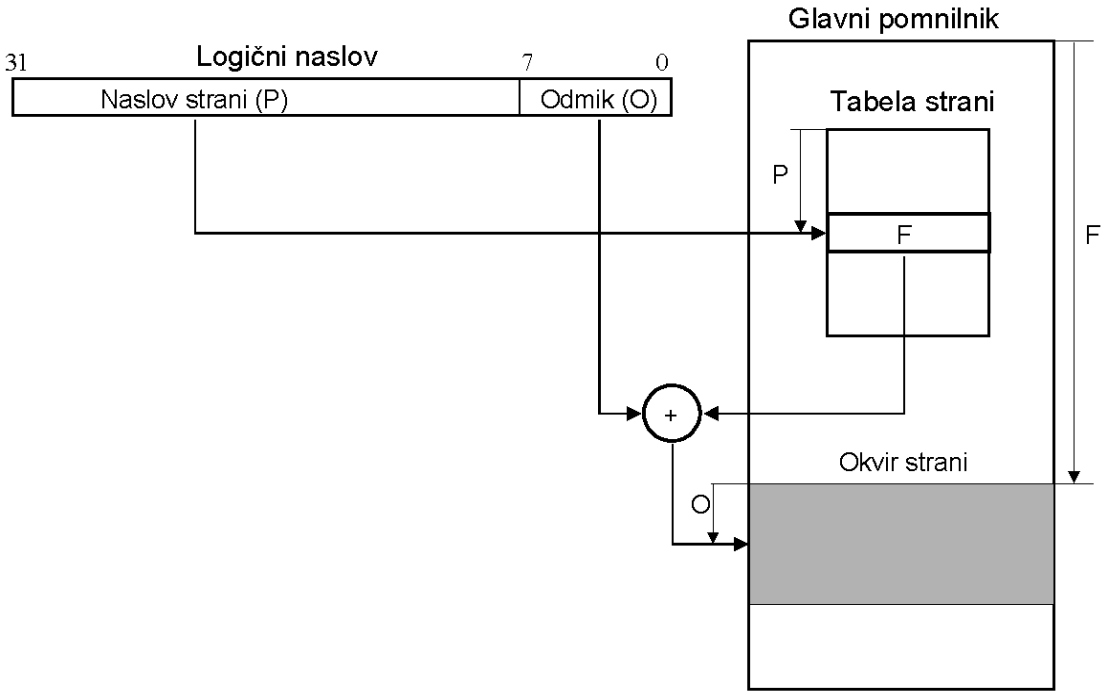


Tabela strani

- Primer:
  - Velikost strani = velikost okvirja = 4KB
  - Logični naslovni prostor: 4 GB, število strani: 1M,
  - Fizični naslovni prostor: 4 MB, število okvirjev 1K.
- Poleg logičnega naslova in fizičnega naslova vsebuje tabela strani še druga določila, bit veljavnosti (V), bit spremembe (M), bit zaščite (WP), bit prisotnosti v pomnilniku, i.t.d.

Tabela strani

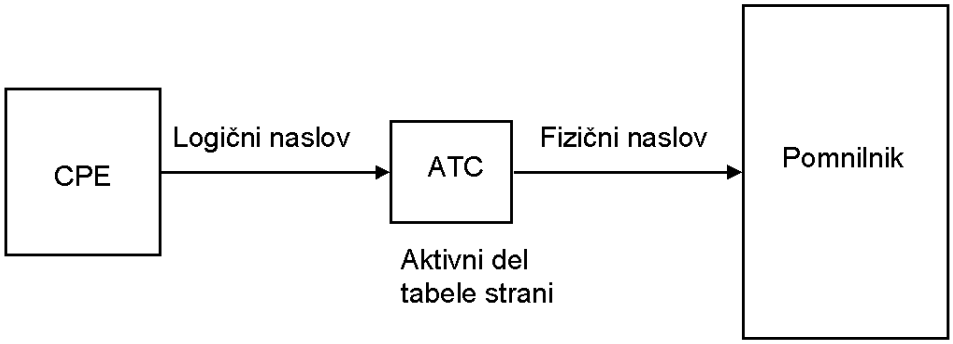
- Tabela strani je lahko zelo velika!
- Primer: Velikost tabele strani = število strani × velikost posamezne komponente tabele = 1 M × 8 = 8 MB.
- Vsak nameščeni program mora imeti svojo tabelo. V večopravilnem sistemu je takih programov več.
- Vprašanje: kam namestiti tabelo (tabele) strani?
- Tabela strani je nameščena v glavnem pomnilniku.
- Posledica: upočasnitev napredovanja programa. Za dano "koristno" (logično) referenco, sta potrebna dva pomnilniška dostopa.



- V uporabi je večnivojska organizacija tabel.
- Vsakemu programu pripada tabela strani prvega nivoja in več tabel drugega, (tretjega, ...) nivoja.
- Za vsako koristno pomnilniško referenco je potrebno toliko dodatnih referenc, kolikor je nivojev tabel.

Naslovni preslikovalni predpomnilnik - ATC

- Preslikava logičnih naslovov v fizične mora biti hitra.
- Tabela strani se nahaja v glavnem pomnilniku.
- Aktivni del tabele strani se nahaja v naslovnem preslikovalnem predpomnilniku, (ATC - Address Translation Cache), (TLB - Translation Look-aside Buffer).
- Ta predpomnilnik je največkrat delno (ali popolnoma) asociativen (4-stransko delno asociativen).
- Dokler je aktivni del tabele strani v predpomnilniku (ATC), se dostop do pomnilnika praktično ne podaljša.
- V nasprotnem primeru je potreben dodaten dostop do pomnilnika (do tabele) strani in polnjenje ATC.
- V primeru, da stran ni v pomnilniku, je potreben dostop do diska.



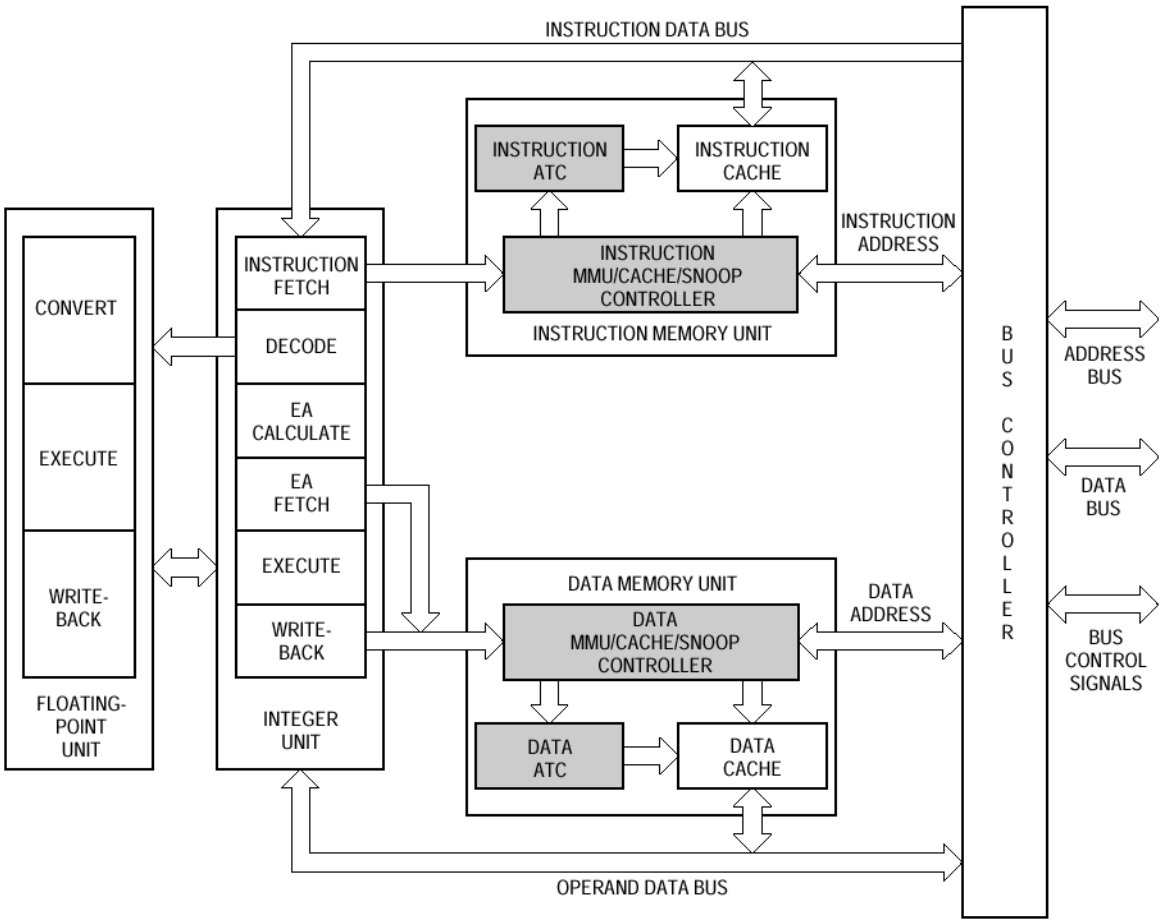
Poleg logičnega naslova in fizičnega naslova vsebuje ATC še določila, bit veljavnosti (V), bit spremembe (M), bit zaščite (WP), bit prisotnosti v pomnilniku, i.t.d.

MC68030 in navidezni pomnilnik

- Dodeljevanje pomnilnika po straneh na zahtevo (ang. Demand Paging).
- Velikost strani: programsko nastavljiva: $n \times 256$ bajtov, (min = 256 bajtov, max = 32 Kbajtov).
- Večnivojska organizacija tabela strani (1 do 5 nivojski sistem tabel).
- Tabele strani se nahajajo v glavnem pomnilniku.
- URP, SRP: (User Root Pointer, System Root Pointer) registra začetnega naslova tabele strani prvega nivoja.
- Prva tabela strani in tabele strani vmesnih nivojev "kažejo" na tabele strani naslednjega nivoja.
- Zadnji nivo tabel "kaže" na okvirje strani.
- Za preslikavo logičnega naslova v fizični naslov je potrebno do šest pomnilniških referenc.
- ATC: (popolnoma) asociativen, hrani do 22 aktivnih preslikav logičnega naslova strani v fizični naslov okvirja + dodatna določila.
- Deklarirana verjetnost zadetka: 98 odstotna.

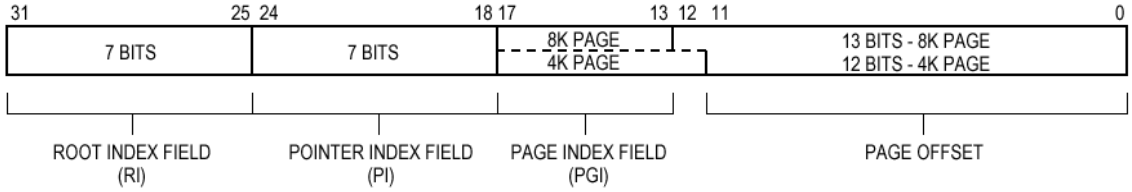
MC68040 in navidezni pomnilnik

- Dodeljevanje pomnilnika po straneh na zahtevo (ang. Demand Paging).
- Dve enoti za upravljanje pomnilnika: ukazna in podatkovna MMU (ang. Memory Management Unit).
- Velikost strani: programsko nastavljiva: 4 Kb ali 8 Kb.



MC68040 in navidezni pomnilnik - sistem tabel

- Trinivojska drevesna organizacija tabel strani, vmesne tabele "kažejo" na tabele naslednjega nivoja, tabele zadnjega nivoja "kažejo" na okvirje strani.



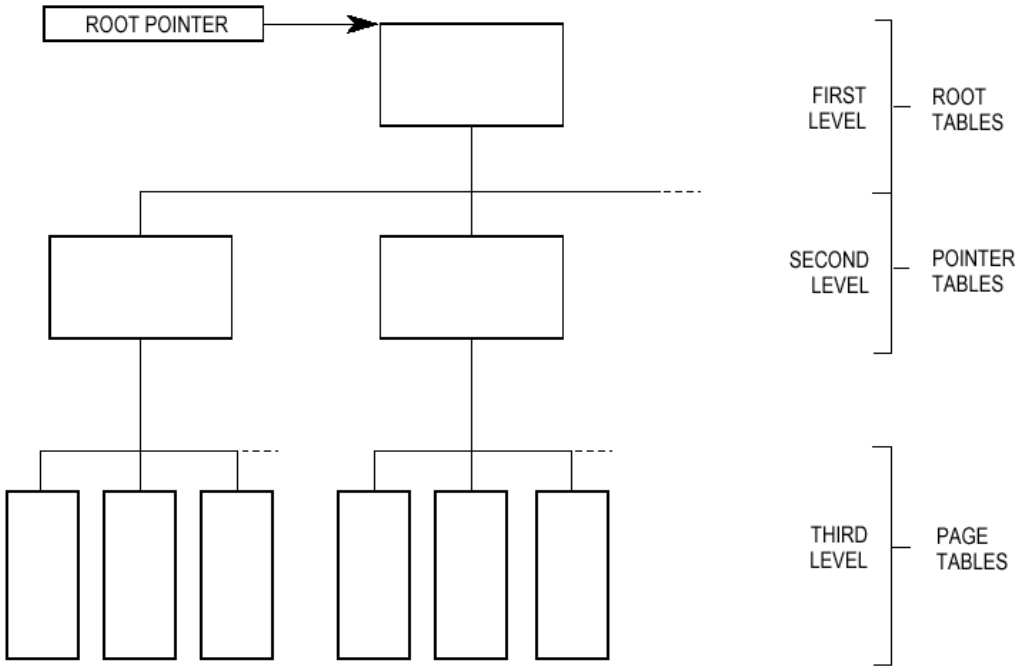
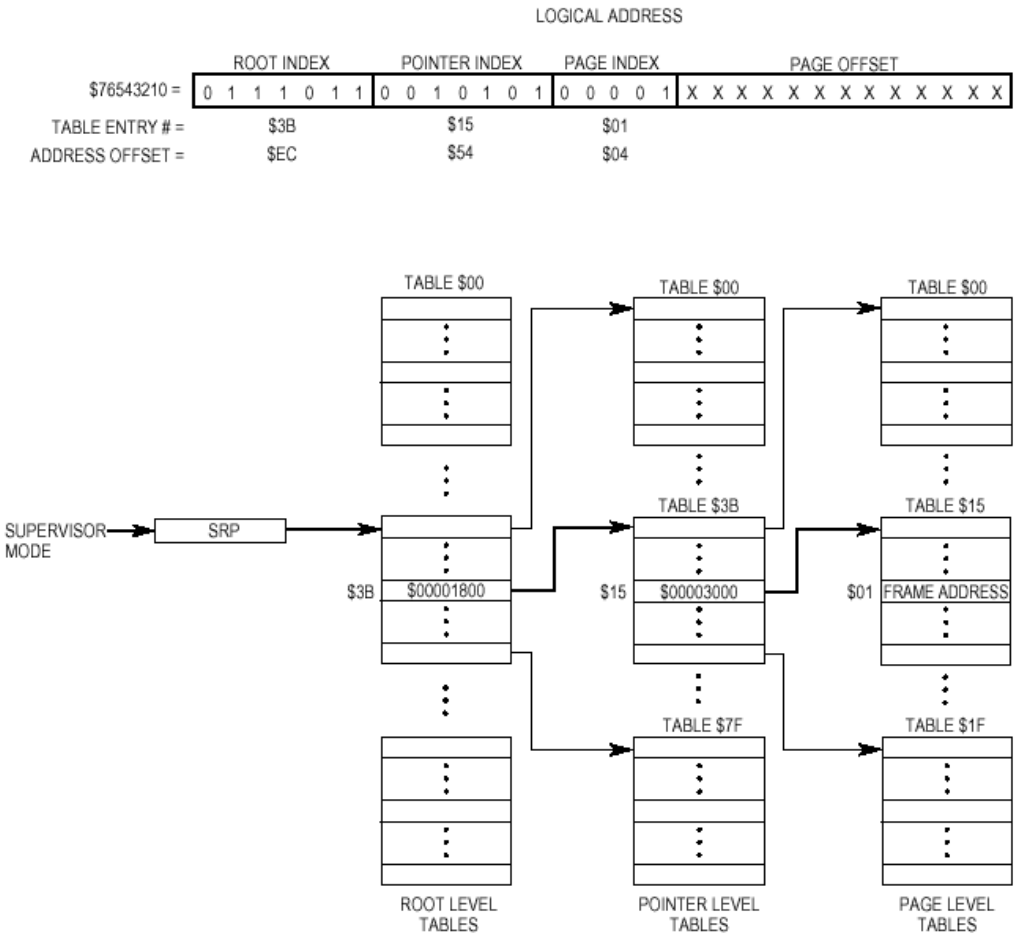
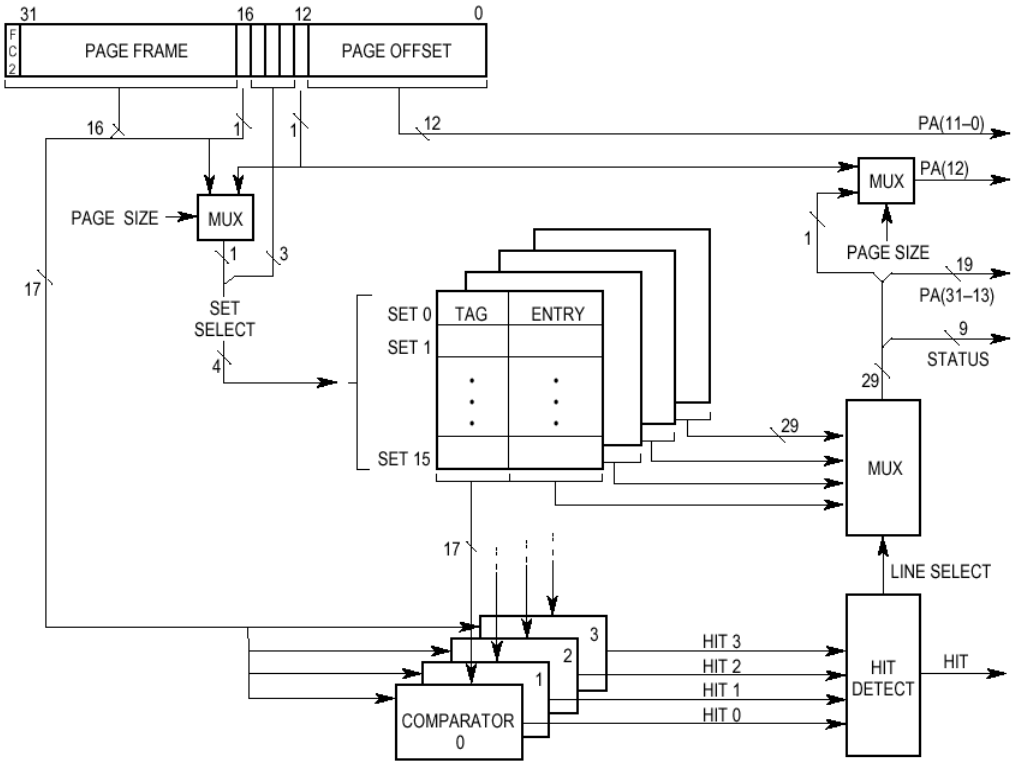


Figure 4-6. Translation Table Structure

MC68040 in navidezni pomnilnik - primer



MC68040 in navidezni pomnilnik - ATC

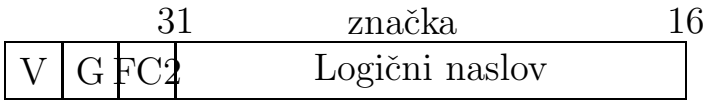


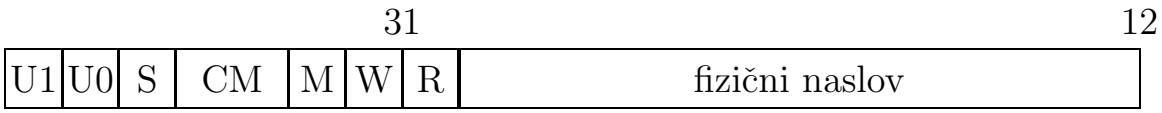
- ATC: 4-stransko delno asociativen, sposoben zabeležiti do 64 aktivnih strani: (16 x 4), za vsako MMU. Polnjenje predpomnilnika je psevdo naključno.
- Registri MMU:
  - URP, SRP: registra začetnega naslova tabele strani prvega nivoja.
  - DTT0, DTT1, ITT0, ITT1: Data/Instruction Transparent Translation Register. (4 pomnilniški bloki z direktno preslikavo logičnih v fizične naslove). (16 Mb).
  - MMUSR: MMU Status register
  - TC: Translation Control Register (Izbira velikosti strani, omogoči/prepreči preslikavo l.n. v f.n.).
- Verjetnost zadetka v ATC je več kot 98 %.

MC68040 - Izgled vsebine v ATC

Stanje ATC se osvežuje iz tabel strani, v odvisnosti od poteka pomnilniške reference.

Oblika vpisa v ATC:





- V: (Valid), bit veljavnosti (veljavna preslikava).
- G: (Global), element globalnega pomena.
- FC2: (Function Code 2), razlikuje uporabniške od sistemskih referenc.
- Logični naslov: 16 bitna naslovna značka.
- U0, U1: (User Page Attributes), definirani s strani uporabnika.
- S: (Supervisor), dostop dovoljen samo v nadzornem načinu.
- CM (Cache mode), določa pomen predpomnilnika na nivoju posamezne strani (vpis skozi, vpis nazaj, brez predpomnilnika).
- M: (Modified), sprememba strani (se postavi ob operaciji vpisa).
- W: (Write Protect), zaščita pred vpisom.
- R: (Resident), se postavi v primeru, da pomnilniška referene konča brez prekinitve (Stran je v pomnilniku).
- Fizični naslov, naslov okvirja strani.

Prekrivanje strani

Vprašanje: katero stran prekriti, ko ni prostega okvirja strani.

- Princip optimalnosti
  - prekrijemo stran, ki ne bo več potrebna.
  - to je v praksi neizvedljivo.
- Na osnovi preteklosti sklepamo na prihodnost.
  - Prekrijemo stran, ki je bila najdlje v pomnilniku.
  - LRU: (Least Recently Used) prekrijemo stran, ki je bila najdlje neuporabljena.
  - NUR: (Not Used Recently) prekrijemo stran, ki ni bila v zadnjem času uporabljena.
  - LFU (Least Frequently Used) prekrijemo stran, ki je bila najmankrat uporabljena.
  - PFF (Page Fault Frequency) prekrivamo na podlagi števila napak strani.