# Memoria virtuale e ndërmjetme - MVN (ang. Middle virtual memory)

## Abstrakt

Sistemet e mbjella zakonisht pranojnë të dhëna nga senzorë, pastaj i përpunojnë ato të dhëna dhe i ruajnë në memorien e tyre.

Për arsye të ndryshme, qoftë organizative apo ekonomike, procesorët kanë memorie të brendshme të vogël RAM dhe memorie pak më të madhe ROM. Andaj, për ruajtje të të dhënave në një sasi më të mëdha përdorim memorie të jashtme.

Memoriet e jashtme mund të jenë flesh memorie apo EEPROM, të cilat komunikojnë përmes protokolleve të ndryshme si I2C ose SPI.

Secila memorie e jashtme ka kohën e qasjes të caktuar text dhe si e tillë është më e madhe sesa koha e qasjes së memories së brendshme RAM.

Andaj, në vend se të përdoret drejtpërdrejtë memoria e jashtme, ne do ta përdorim një pjesë të memories RAM si memorie virtuale e ndërmjetme.

Puna e kësaj memorie do të jetë analoge me punën e memories kesh, ashtu që një pjesë e të dhënave të memories së jashtme do të bartet në këtë memorie virtuale të ndërmjetme.

Është me rëndësi të ceket se kjo memorie nuk kërkon ndërhyrje fizike në procesor apo jashtë tij, thjeshtë duhet që në mënyrë programore të rezervohet kjo hapësirë e memories.

## Hyrje

Në sistemet e mbjella (ang. embedded systems), sikurse në çdo sistem tjetër, memoria e brendshme RAM nuk i përmbushë kërkesat për ruajtje dhe operim me një sasi të madhe të të dhënave.

Andaj, detyrimisht për ruajtje të të dhënave në sasi më të mëdha duhet të përdoren memoriet e jashtme. Varësisht nga projekti dhe të dhënat që do t’i ruajmë, mund të përdorim numër të ndryshëm të memorieve të jashtme me kapacitet të ndryshëm.

Memoriet e jashtme mund të jenë të llojeve të ndryshme por në përgjithësi hasen ato të llojit Flash dhe EEPROM, të cilat punojnë me ndërfaqe (ang. interface) të ndryshme, por ne kemi përdorur SPI.

Dallimi esencial në mes tyre është se në memorien flash mund të shkruhet në një adresë të saj vetëm nëse ajo paraprakisht ka qenë e fshirë, pra ka qenë me vlerë të nënkuptueshme 0x00 ose 0xFF; ndërsa në memorien eeprom mund të shkruajmë në një adresë të caktuar, pavarësisht se a ka qenë e fshirë paraprakisht apo jo dhe çfarë vlere përmban ajo. Tjetër dallim shumë i rëndësishëm ndërmjet tyre është se në memorien flash nuk mund të fshihet përmbajtja e vetëm një adrese mirëpo fshirja duhet të bëhet në bllok, pra më shumë adresa, ndërsa në memorien eeprom fshirja bëhet duke e rishkruar përmbajtjen e adresës së caktuar me një vlerë të nënkuptuar 0x00 ose 0xFF.

Duke pas parasysh këtë, në këtë hulumtim do t’i përdorim të dy llojet e memorieve, me të njëjtën ndërfaqe dhe do ta shohim dallimin dhe përparësitë ndaj njëra-tjetrës, por përfundimisht gjatë gjithë projektit si memorie të jashtme do ta përdorim memorien eeprom me ndërfaqe spi.

Duke marrë parasysh se memoria e jashtme ka shpejtësi të qasjes dukshëm më të vogël (text = 25 – 50 us) sesa memoria RAM (tram 50-100 ns), atëherë është e qartë se aplikacioni do të punojë më shpejtë nëse të dhënat janë në RAM sesa në memorie të jashtme.

Andaj, ne propozojmë që një pjesë e RAM memories të luaj rolin e memories kesh për memorien e jashtme, pra t’i ketë disa korniza të të dhënave të memories së jashtme dhe kësisoj të përshpejtojmë qasjen në to. Këtë memorie e kemi quajtur memorie virtuale të ndërmjetme ose shkurt mvn.

Termi “virtual” është përdorur për arsye se në të vërtetë kjo memorie nuk është memorie fizike mirëpo është memorie vetëm e definuar në mënyrë programore. Ndërsa termi “e ndërmjetme” është përdorur për arsye se luan rolin e ndërmjetësit në mes të mikrokontrollerit dhe memories së jashtme.

Shpejtësia e qasjes në mvn tmvn do të jetë sa shpejtësia e qasjes në RAM tram , duke i shtuar në shpejtësi të procedimit të të dhënave Δt. Pra, tmvn ≈ tram

## Algoritmi i kërkimit të të dhënës

Në rastin kur një e dhënë i duhet procesorit, ai së pari duhet të kërkojë në memorien virtuale të ndërmjetme. Nëse nuk e dhëna nuk gjendet në atë memorie atëherë procesori duhet të kaloj tek memoria e jashtme dhe ta kërkojë atë, pastaj nëse e dhëna gjendet në memorie të jashtme duhet që ajo të sillet në procesor. Përveç të dhënës, sillen edhe një sasi K e kornizave tjera që mund të kërkohen më vonë.

Në fillimin e punës së procesorit, memoria virtuale e ndërmjetme mbushet me të dhëna nga memoria e jashtme duke filluar nga adresa e parë e saj.

Pra, algoritmi i kërkimit të të dhënës është:

1. Kërko në memorien virtuale të ndërmjetme.
2. Nëse e dhëna gjendet në memorien virtuale të ndërmjetme, atëherë kemi një qëllim (ang. hit).
3. Nëse e dhëna nuk gjendet në memorie virtuale të ndërmjetme, atëherë kemi një mungesë (ang. miss), dhe e dhëna kërkohet në memorie të jashtme.
4. Nëse e dhëna gjendet në memorie të jashtme:
   1. Procedo të dhënën
   2. Bartë K korniza (ang. frame) në memorie virtuale të ndërmjetme. Ashtu që, barten K/2 korniza para të dhënës dhe K/2 pas të dhënës si dhe vet korniza e të dhënës. Pra, barten gjithsej K+1 korniza.

Nëse procesori përkrahë teknologjinë DMA atëherë të dhënat do të barten përmes kësaj teknologjie.



## Hartëzimi

Madhësia e memories virtuale së ndërmjetme duhet të jetë e plotpjesëtueshme me madhësinë e kornizës që ndodhet në memorie të jashtme.

Në rastin tonë madhësia e një kornize në memorie të jashtme është 5 bajt, në të cilën përmbahen numri i kartelës (3bajt) dhe qasja (bajt). Pra, në këtë rast memoria virtuale e ndërmjetme duhet të mund t’i përmbajë (K+1) \* 5 bajt.

Pra, memoria MVN është e ndarë në NUMBER\_OF\_STRUCTURES struktura prej SIZE\_OF\_STRUKTURE bajtësh, që në rastin tonë është 5 bajtësh.

Kur e korniza e kërkuar nga procesori gjendet në memorie virtuale të ndërmjetme, atëherë koha e qasjes është tmvn (50-100 ns).

Mirëpo kur korniza nuk gjendet në mvn, ajo duhet të kërkohet në memorie të jashtme dhe kështu rritet vonesa në procedim të të dhënave dhe koha e qasjes së tyre është text .

Kur një e dhënë nuk gjendet në memorie virtuale të ndërmjetme, atëherë ajo duhet të sjellët nga memoria e jashtme. Për ta bërë më të lehtë dhe më të shpejtë përgjigjen ndaj kërkesës për të dhënat, kur e dhëna e kërkuar nga procesori gjendet në memorie të jashtme, ajo nuk do të bartet menjëherë në mvn. Procesori njëherë e procedon atë të dhënë dhe tek pasi të ketë përfunduar punë me të përgatitet për bartje të kornizave në mvn.

Koha për të cilën të dhënat barten nga memoria e jashtme në mvn është: tc = text + tmvn + Δt. Koha Δt paraqet kohën e cila i nevojitet procesorit t’i procesorit për procedimin e moduleve, koha e bartjeve etj.

Se ku do të vendoset korniza në mvn, sikurse në memorien kesh, edhe këtu kemi disa teknika që mund të përdoren për këtë qëllim:

1. Hartëzimi direkt
2. Hartëzimi i shoqërimit në blloqe
3. Hartëzimi i shoqërimit të plotë

Se cilën teknikë do të përdorim, do të definohet përmes një direktive të definuar si MAPPING\_TECHNIQUE.

### Hartëzimi direkt

Siç përmendëm më lartë, memoria MVN përmban NUMBER\_OF\_STRUCTURES struktura, secili me madhësi sa madhësia e kornizës.

Andaj, në hartëzimin direkt bëhet në këtë mënyrë:

Për shembull, nëse kemi një memorie MVN me 100 blloqe dhe nëse dëshirohet që korniza *k* me adresë 0xAF16 (17510) të bartet në memorien MVN, atëherë adresa në të cilën do të bartet kjo kornizë do të jetë:

Për të definuar hartëzimin në modin direkt, duhet që direktiva MAPPING\_TECHNIQUE të jetë MAPPING\_TECHNIQUE = DIRECT\_MAPPING.

### Hartëzimi i shoqërimit në blloqe

Nëse modin e hartëzimit e zgjedhim të jetë i shoqëruar në blloqe, atëherë memorien MVN duhet ta ndajmë në B blloqe të cilat përmbajnë një numër uniform të strukturave SB.

Në këtë mod të hartëzimit, blloku se ku do të vendoset një kornizë *k* nga memoria e jashtme në memorien MVN bëhet ngjashëm sikurse tek hartëzimi direkt, vetëm se këtu numri i strukturave S zëvendësohet me numrin e blloqeve B.

Pas llogaritjes së bllokut, gjejmë edhe adresën e strukturës brenda bllokut në të cilën do të vendoset korniza. Këtë adresë e gjejmë duke gjetur modin ndërmjet bllokut të zgjedhur dhe numrit të strukturave në bllok, pra:

Shembull: Nëse kemi memorie MVN me 10 blloqe (B = 10), ku secili prej blloqeve ka 10 struktura (SB = 10), dhe korniza në RAM memorie me adresë 0xAF16 (17510).

Atëherë do të kemi:

Pra, kjo kornizë do të vendoset në bllokun 5, në strukturën 5.

Ky mod i hartëzimit zgjedhet duke barazuar MAPPING\_TECHNIQUE = SET\_MAPPING.

### Hartëzimi i shoqërimit të plotë

Në këtë mod të hartëzimit secila kornizë e memories së jashtme mund të bartet në secilën strukturë të memories MVN.

Së pari do të shikohet nëse ndonjë strukturë e memories MVN është e lirë dhe nëse gjendet ndonjëra atëherë korniza bartet në atë strukturë, mirëpo nëse nuk ka strukturë të lirë atëherë duhet të aplikohet ndonjë teknikë e zëvendësimit. Ky mod i hartëzimit zgjedhët duhet barazuar MAPPING\_TECHNIQUE = FULL\_MAPPING.

## Teknikat e zëvendësimit

Më lartë treguam se në modin e hartëzimin e shoqërimit të plotë, duhet që të përdoren teknika të zëvendësimit. Ne do ta bëjmë të mundur që të zgjedhët një teknikë e dëshiruar, nga një listë që do ta paracaktojmë.

Zgjedhja e kësaj teknike bëhet duke barazuar direktivën REPLACE\_TECHNIQUE me direktivën e teknikës së dëshiruar.

Më poshtë po tregojmë teknikat që mund të zgjedhën si dhe si ato zbatohen në praktikë.

### FIFO (ang. First-In First-Out)

Teknika e zëvendësimit FIFO (ang. First-In First-Out) është teknikë relativisht e thjesht, në të cilën korniza që hynë e para del e para, apo zëvendësohet e para. [fifo.pdf]

Për të ruajtur sekuencën e hyrjeve të kornizave në memorie, kemi një numërues i cili ruan adresën/indeksin e strukturës më të vjetër në të cilën ka hyrë ndonjë korniza më e vjetër.

Kështu, kur një strukturë duhet të zëvendësohet, së pari gjejmë adresën e strukturës më të vjetër (duke lexuar vlerën e numëruesit), e pastaj në memorien MVN zëvendësojmë strukturën me adresën e gjendur.

Kur të dhënat e kërkuara nga procesori gjenden në memorie MVN atëherë nuk kemi nevojë që të përditësojmë numëruesin. Ndërsa në rastin kur të dhënat nuk gjenden në MVN dhe duhet të barten nga memoria e jashtme, atëherë para secilës bartje të kornizës duhet të përditësojmë numëruesin[fifo.pdf] duke rritur për 1 vlerën e tij, duke pasur kujdes të mos tejkalohet adresa e fundit të strukturave:

void updateSequenceCounter()

{

sequenceCounter =

/\*

kontrollo nëse vlera e ardhshme e numruesit të sekuencës është më e madhe sesa numri total i strukturave

\*/

(sequenceCounter+1 > NUMBER\_OF\_STRUCTURES])

?

0 // nëse është më i madh, atëherë ktheje në gjendjen zero

:

sequenceCounter + 1; //pëndryshe rrit në vlerën tjetër

}

### LRU (ang. Least Recently Used)

Teknika e zëvendësimit LRU zëvendëson strukturën e cila ka më së paku përdorime. Të gjitha qasjet në strukturë duhet të regjistrohen në një varg, ashtu që indeksi i një anëtari përkon me adresën ose indeksin e strukturës., ndërsa vlera e anëtarit të atij indeksi tregon frekuencën e qasjes në strukturë. Vargu për numërim ka kapacitet sa numri i strukturave në MVN.

Pra, ky varg për numërim duhet të ndryshohet sa herë që qaset një strukturë në memorie MVN.

Kur një kornizë duhet të bartet në memorien MVN atëherë duhet që së pari të gjejmë adresën e strukturës e cila duhet të ndërrohet duke gjetur indeksin e vlerës minimale në vargun për numërim.

int frequncyCounter[NUMBER\_OF\_STRUCTURES];

int LRUIndex(){

int minimal = \*(frequncyCounter + 0);

int i = 0, lruIndex;

for (i = 1; i < NUMBER\_OF\_STRUCTURES; i++){

if(frequncyCounter[i] < minimal){

minimal = frequncyCounter[i];

lruIndex = i;

}

}

return lruIndex;

}