Universiteti i Prishtinës "Hasan Prishtina" Kosovë

Arkitektura e kompjuterit

Pjesa 6 – Memoria

Prof. Asoc. Dr. Ermir Rogova

Objektivat

- Të zotërohet kuptimi i hiearkisë së memories
- Të kuptohet si çdo nivel i memories i kontribuon performansës së sistemit, si dhe si matet performansa e sistemit
- Të zotërohen kuptimet e memories "cache", të memories virtuale, të segmentimit të memories, të faqosjes (paging) dhe të përkthimit të adresave

Hyrje

- Memoria është në qendër të kompjuterit që ruan programin
- Në kapitujt e mëhershëm studiuam komponentat prej të cilave është e përbërë memoria, si dhe mënyrat se si bashkësitë e ndryshme të instruksioneve i qasen memories
- Këtu do të përqendrohemi në çështje të organizimit
- Të kuptuarit e qartë e këtyre ideve është me rëndësi fundamentale për performansën e sistemit

Llojet e memories

- Janë dy lloje të memories qëndrore: memoria me qasje direkte (RAM) dhe memoria-vetëm-për-lexim (ROM)
- Janë dy tipe të memories RAM, RAM-i dinamik (DRAM) dhe RAM-i statik (SRAM)
- DRAM përbëhet prej kapacitorëve, të cilët ngadalë e humbin ngarkesën e tyre, prandaj ata duhet të rimbushen brenda disa milisekondave për ta penguar humbjen e të dhënave
- Për shkak të dizanjit të vet DRAM është një memorie e lirë

Llojet e memories

- SRAM përbëhet prej qarqeve të ngjashme me flip-flopin D nga kapituli 3, prandaj qelizat e memories e ruajnë përmbajtjen e tyre deri sa ka rrymë dhe nuk kanë nevojë të rimbushen
- SRAM është memorie shumë më e shpejtë se DRAM, por çipat e DRAM kanë densitet më të madh, prandaj mund të mbajnë më shumë informata se çipat e SRAM, ndërkohë që memoria SRAM nxehet shumë më tepër dhe shpenzon më shumë energji se DRAM
- Për këtë arsye teknologjia DRAM përdoret për ndërtimin e memories RAM, ndërkaq teknologjia SRAM për ndërtimin e memories "cache"

Llojet e memories

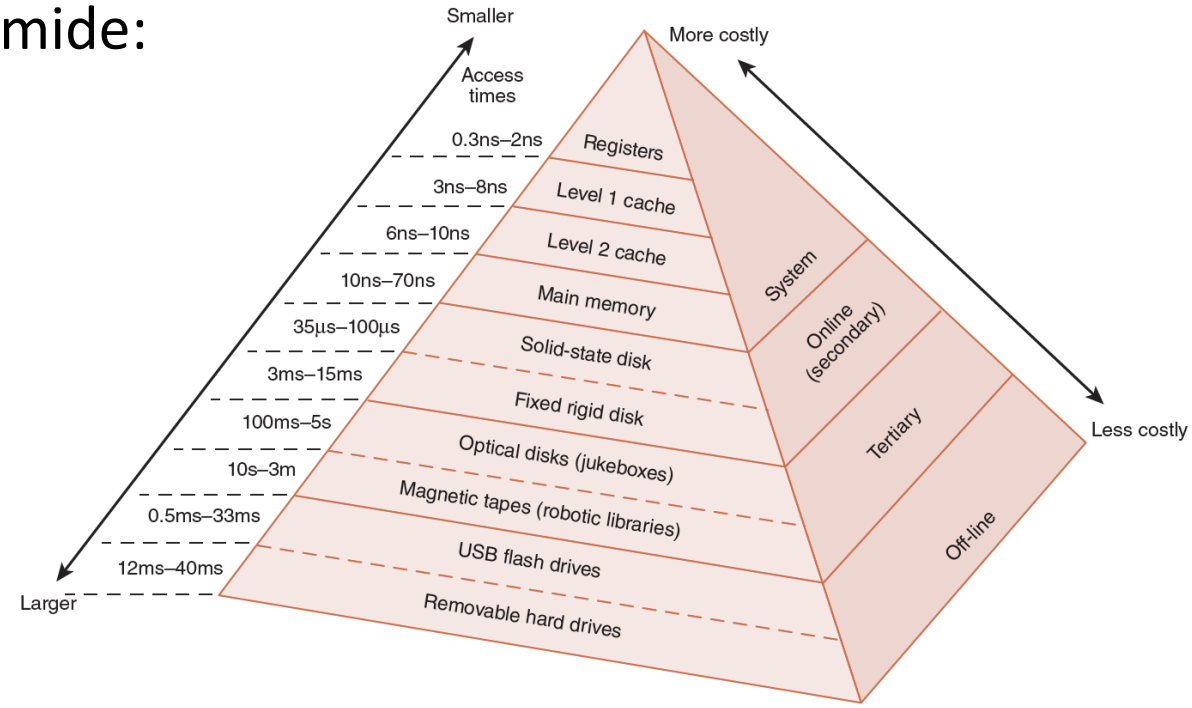
- Memoria ROM nuk ka nevojë për rimbushje. Asaj i nevojitet shumë pak energji për ta ruajtur përmbajtjen
- ROM përdoret për ruajtjen e të dhënave të përhershme ose gjysëm të përhershme, të cilat ruhen edhe kur sistemi është i fikur
- Llojet e ROM
  - PROM
  - EPROM
  - EEPROM
  - Flash memoria

Hierarkia e memories

- Në përgjithësi, memoria më e shpejtë është më e shtrenjtë se sa memoria më e ngadalshme
- Për të ofruar performansë më të mirë me çmim sa më të ulët memoria organizohet në formë hierarkie
- Elementet me kapacitet më të vogël dhe shpejtësi më të madhe ndodhen brenda ose afër CPU-së, përderisa memoria më e ngadalshme dhe me kapacitet më të madh mund të arrihet përmes magjistralës
- Një kapacitet edhe më i madh i memories, në formë të disqeve dhe shiritave magnetikë ndodhet edhe më larg nga CPU

Hierarkia e memories

- Organizimi i memories mund të paramendohet në fomë piramide:



Hierarkia e memories

- Për t'iu qasur një të dhëne konkrete CPU së pari dërgon një kërkesë memories më të afërt, kryesisht "cache"
- Nëse e dhëna e kërkuar nuk është në "cache", atëherë kërkohet në memorien qëndrore.
- Nëse nuk është as në memorien qëndrore, kërkesa i drejtohet diskut
- Në momentin kur përcaktohet lokacioni i të dhënës së kërkuar, e njëjta, si dhe elementet pranë saj barten në memorien "cache"

Hierarkia e memories

- Disa përkufizime:
  - Hit – e dhëna është gjetur në nivelin e caktuar të memories.
  - Miss – e dhëna nuk është gjetur në nivelin e caktuar.
  - Hit rate – sa herë është gjetur e dhëna në nivelin e caktuar të memories (përqindje)
  - Miss rate – sa herë NUK është gjetur e dhëna në nivelin e caktuar të memories (përqindje)
  - Miss rate = 1 - Hit rate.
  - The hit time – koha e nevojshme për t'iu qasur të dhënave në nivelin e dhënë të memories
  - The miss penalty – koha e nevojshme për të përpunuar një miss, përfshirë edhe kohën e nevojshme për ta zëvendësuar një bllok të memories plus koha e nevojshme për t'i dërguar të dhënat në procesor.

Hierarkia e memories

- Pas çdo hit-i një bllok i tërë i të dhënave kopjohet nga memoria, pasiqë parimi i lokalitetit na tregon se nëse i qasemi një bajti është shumë e mundshme se të dhënat fqinje do të nevojiten së shpejti.
- Dallojmë tre forma të lokalitetit:
  - Lokaliteti kohor – të dhënave, të cilave iu kemi qasur së voni, ka gjasa t'i qasemi përsëri në një të ardhme të afërt
  - Lokaliteti hapësinor – urdhërat për qasje memories grupohen në hapësirën adresore (p.sh. në formë të matricave).
  - Lokaliteti sekuencial – ekziston tendenca që instruksioneve t'i qasemi në formë sekuenciale.

Memoria "Cache"

- Në kompjuterët personalë madhësia tipike e memories " L3 Cache" sillet në mes të 2MB dhe 256 MB, " L2 Cache" sillet në mes të 64 KB dhe 2 MB, ndërkohë që madhësia tipike e "L1 Cache" sillet prej 8-64 KB
- Kompjuteri nuk ka asnjë mundësi të dijë se cila bashkësi e të dhënave do të përdoret, prandaj, në bazë të parimit të lokalitetit, bëhet transferi i një blloku të tërë nga RAM në "Cache".
- Lokacioni i këtij blloku në "cache" varet nga madhësia e "cache" dhe nga rregullat e pasqyrimit të "cache"-it.

Memoria "Cache"

- Qëllimi i memories "cache" është të përshpejtojë qasjen duke i ruajtur të dhënat më afër CPU-së, në vend se t'i ruajë në memorien qëndrore
- Edhepse "cache" është shumë më i vogël se memoria qëndrore, koha e qasjes është shumë më e shkurtër
- Për dallim nga memoria qëndrore, të cilës i qasemi përmes adresës, memories "cache" i qasemi përmes përmbajtjes, prandaj edhe quhet memorie e adresueshme sipas përmbajtjes (MAP)
- Për shkak të kësaj, nuk është e dëshirueshme që memoria "cache" të jetë e vetme dhe me kapacitet të madh, sepse humbet shumë kohë për kërkim

Memoria "Cache"

- Përmbajtja që adresohet në MAP është nënbashkësi e bitëve të një adrese të memories qendrore, e cila quhet fushë
- Fushat në të cilat ndahet adresa e memories paraqesin një pasqyrim "mbi" nga memoria e madhe RAM në memorien më të vogël "cache"
- Shumë blloqe të memories qëndrore pasqyrohen në një bllok të "cache". Ndërkaq, fusha tag (etiketë) në bllokun e "cache" bën dallim në mes të blloqeve të ndryshme të memories

Memoria "Cache"

- Ekzistojnë skema të ndryshme të pasqyrimit nga memoria në "cache"
- Para të gjithash, memoria kryeore dhe "cache" ndahen në blloqe të madhësisë së njëjtë
- Kur të gjenerohet një adresë e memories, së pari bëhet kërkimi në "cache" i fjalës përkatëse
- Nëse fjala e kërkuar nuk gjendet në "cache", atëherë i tërë blloku nga RAM bartet në një bllok të "cache"

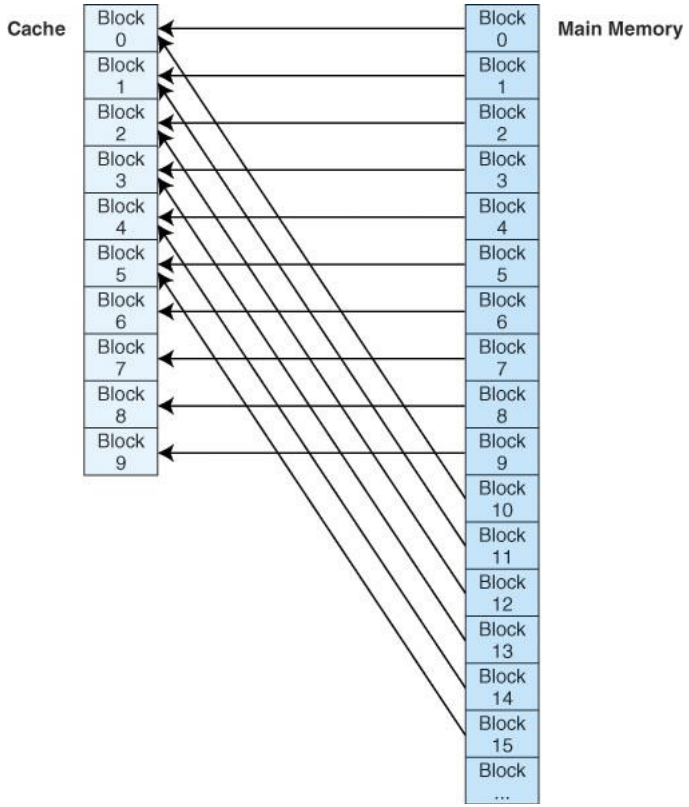
Memoria "Cache"

- Si përdoren fushat në adresat e memories kryesore?
  - Një fushë e adresës tregon lokacionin në "cache" ku ndodhet fjala e kërkuar (cache hit) ose ku duhet të vendoset nëse nuk është në "cache" (cache miss).
  - Blloku i referuar i "cache" vërtetohet për të konstatuar se a është valid, përmes bitit të validitetit që i shoqërohet çdo blloku(nëse ky bit është 0, atëherë blloku nuk është valid (cache miss) dhe duhet t'i qasemi RAM, ndërsa nëse është 1, blloku është valid (cache hit).
  - Krahasohet fusha tag e "cache" me fushën tag të RAM dhe nëse përputhen, atëherë e kemi gjetur bllokun.
  - Mbetet të gjendet lokacioni i fjalës brenda bllokut.

Memoria "Cache"

- Skema më e thjeshtë e pasqyrimit të "cache" është pasqyrimi i drejtëpërdrejtë i "cache"
- Në këtë skemë të përbërë prej N blloqeve të "cache", blloku X i memories qëndrore pasqyrohet në bllokun e "cache" Y = X mod N
- Pra nëse i kemi 10 bloqe të "cache", blloku 7 i "cache" mund t'i mbajë blloqet 7,17,27,37,... të RAM
- Kur blloku i memories kopjohet në bllokun përkatës në "cache", biti i validitetit vendoset në 1 për t'i treguar sistemit se blloku përmban të dhëna valide

Memoria “Cache”



Memoria "Cache"

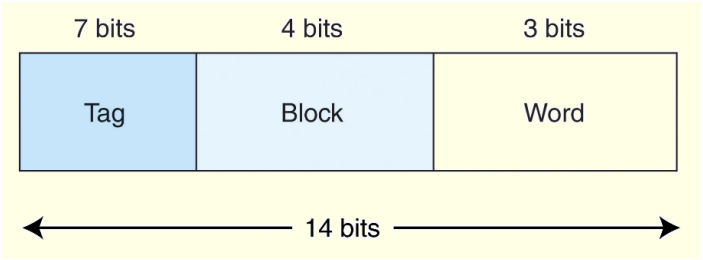
- Kjo është skema e "cache".

Block	Tag	Data	Valid
0	00000000	words A, B, C,...	1
1	11110101	words L, M, N,...	1
2	----------		0
3	----------		0

- Blloku 0 përmban disa falë nga memoria identifikuar me tagun 00000000, ndërsa blloku 1 përmban fjalë të identifikuara me tagun 11110101
- Dy blloqe të tjera janë jovalide

Memoria "Cache"

- Madhësia e fushave të adresës së memories varet nga madhësia e "cache"-it.
- Supozojmë se memoria ka 2^{14} fjalë, "cache" ka $16 = 2^4$ blloqe, dhe cdo bllok përmban 8 fjalë.
  - Pra, memoria është e ndarë në $2^{14} / 2^3 = 2^{11}$ blloqe.
- Adresa e memories është 14-bitëshe. Madhësia e fushave: 4 bits për fushën e bllokut (16 blloqe), 3 bit për fushën e fjalës (8 fjalë), dhe 7 bit për fushën e tagut:

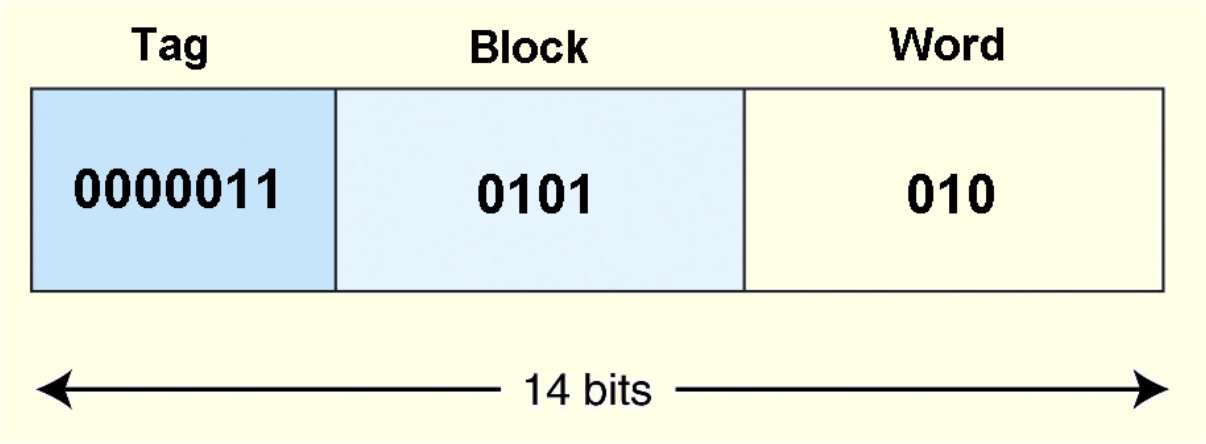


Memoria "Cache"

- Supozojmë se programi gjeneron adresën e memories 1AA, në formë 14-bitëshe 00000110101010
- 1AA = 426 : 8 =53 (mbetja 2), që d.m.th. se adresa 1AA ndohet në bllokun 53 të RAM
- Ky bllok i RAM pasqyrohet në bllokun e "cache" 5 = 53 mod 16
- Ky fakt mund të konstatohet edhe drejtpërdejt nga analiza e vargut 00000110101010

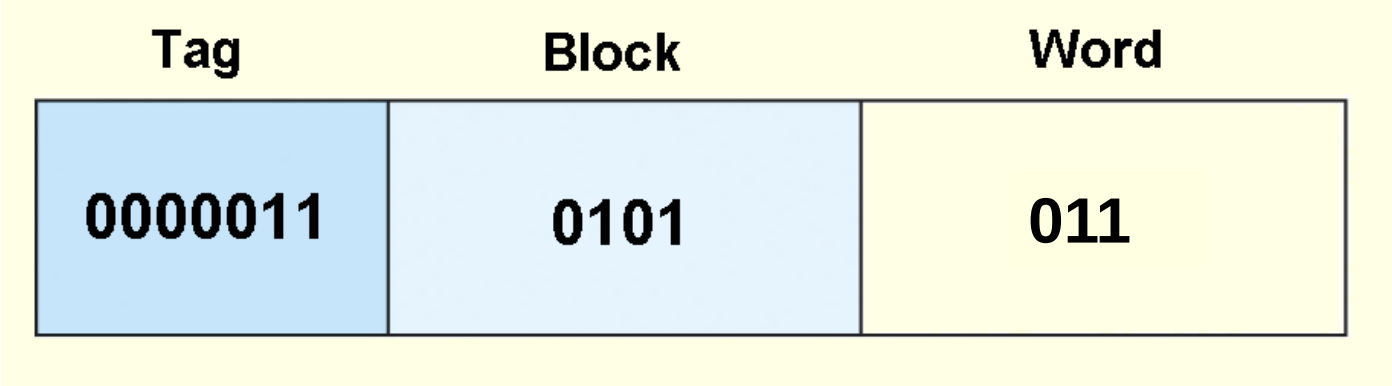
Memoria "Cache"

- 1AA = 00000110101010
- 0101 – blloku 5 i "cache"
- 00000110101 – blloku 53 i RAM



Memoria "Cache"

- Nëse më vonë programi gjeneron adresën 1AB, në formë binare 0000011010101l, e dhëna do të gjindet në bllokun e njëjtë, 0101, fjala 011.



- Ndërkaq, nëse blloku gjeneron adresën 3AB=939 (00010110101011), atëherë blloku 53 i RAM do të hiqet nga bloku 5 i "cache" dhe do të zëvendësohet me bllokun 117 =00010110101 të RAM ku ndodhet adresa 3AB.

Memoria "Cache"

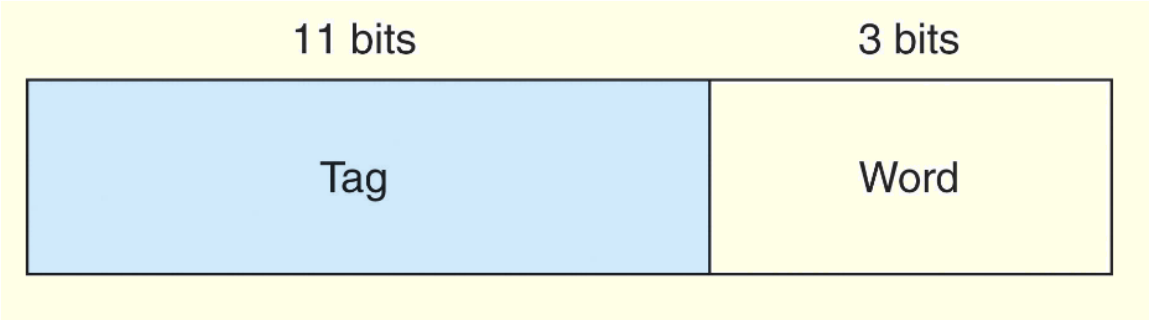
- Supozojmë se programi gjeneron varg të adresave si 1AB, 3AB, 1AB, 3AB, . . . "Cache" do t'i zëvendësojë vazhdimisht blloqet
- Avantazhi teorik që ofron "cache" është humbur në këtë rast ekstrem
- Kjo është mangësia kryesore e "cache"-it të pasqyruar në mënyrë të drejtëpërdrejtë
- Skema të tjera pasqyrimi janë disenjuar për ta penguar këtë lloj çrregullimi

Memoria "Cache"

- Në vend se ta vendosim bllokun e memories në një lokacion të caktuar në "cache", varësisht nga adresa e tij, mund të lejojmë që blloku i memories të vendoset kudo në "cache"
- Kështu, "cache" do të mund të mbushet para se të bëhet fshirja e ndonjë blloku
- Kështu funksionon "fully associative cache"
- Adresa e memories ndahet vetëm në dy pjesë: tag dhe fjala

6.4 Memoria "Cache"

- Si më parë, supozojmë se adresat e memories janë 14-bitëshe, ndërsa "cache" përbëhet prej 16 blloqeve të përbëra prej 8 fjalësh. Formati i fushës për lokacionin e memories është:



- Kur bëhet kërkimi në "cache", paralelisht kontrollohen të gjithë tagjet, në mënyrë që të dhënat të lexohen sa më shpejtë. Kjo kërkon një harduer më të shtrenjtë.

Memoria "Cache"

- "Cache" i drejtëpërdrejtë e largon bllokun e memories sa herë që bllokut tjetër i nevojitet hapësira
- Te "cache" asociativ nuk kemi pasqyrim të tillë, prandaj duhet gjetur një algoritëm, i cili përcakton cili bllok fshihet nga "cache"
- Blloku që fshihet quhet viktimë
- Ka mënyra të ndryshme për të zgjedhur viktimën

Memoria "Cache"

- "Set associative cache" kombinon idenë e "direct mapped cache" dhe "fully associative cache"
- Pasqyrimi "set associative cache" N i ngjan "cache"-it të drejtëpërdrejtë, sepse referenca e memories pasqyrohet në lokacionine caktuar në "cache"
- Për dallim nga "cache" i drejtëpërdrejtë, referenca e memories pasqyrohet në një bashkësi të blloqeve, ngjashëm me "fully associative cache"
- Në vend se të pasqyrohet kudoqoftë në "cache", referenca e memories mund të pasqyrohet vetëm në një nënbashkësi të caktuar të blloqeve të "cache"

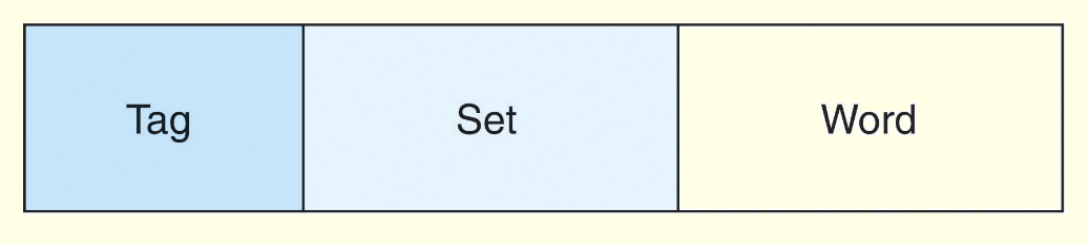
Memoria "Cache"

- Numri i blloqeve të "cache"-it për një bashkësi në "set associative cache" varet nga dizajni i përgjithshëm i sistemit
- P.sh., "set associative cache" mund të konceptohet si në skemën e mëposhtme
- Çdo bashkësi përmban dy blloqe të ndryshme të memories

Set	Tag	Block 0 of set	Valid	Tag	Block 1 of set	Valid
0	00000000	Words A, B, C, . . .	1	----------		0
1	11110101	Words L, M, N, . . .	1	----------		0
2	----------		0	10111011	P, Q, R, . . .	1
3	----------		0	11111100	T, U, V, . . .	1

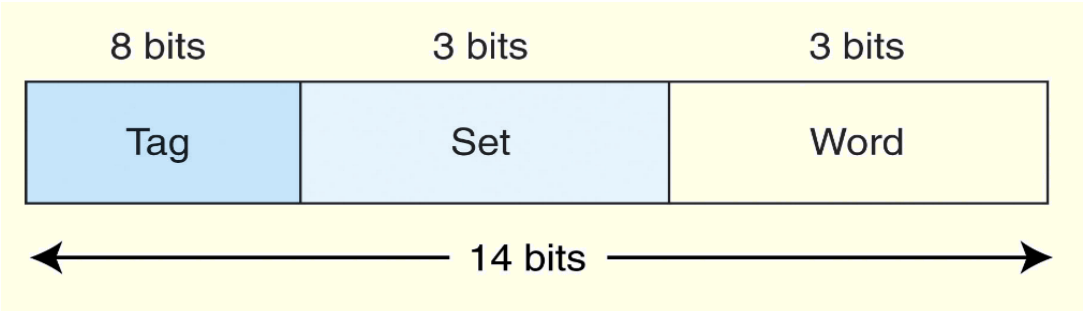
Memoria "Cache"

- Në pasqyrimin "set associative cache", referenca e memories ka tre fusha: tag, set dhe word
- Sikur te "cache" i drejtëpërdrejtë, fusha e fjalës zgjedh fjalën brenda bllokut, ndërsa fusha e tag-ut, identifikon në mënyrë unike adresën e memories
- Fusha set determinon bashkësinë ku bëhet pasqyrimi



Memoria "Cache"

- Supozojmë se kemi memorie qëndrore prej 2^{14} B.
- Kjo memorie pasqyrohet në "2-way set associative cache" me 16 blloqe, ku cdo bllok përmban 8 fjalë.
- Cdo bashkësi i ka 2 blloqe, prandaj kemi 8 bashkësi.
- Pra, nevojiten 3 bit për bashkësi, 3 bit për fjalë, ndërsa 8 bitët e mbetura shfrytezohen për tag.



Memoria "Cache"

- Tek të dy llojet e përmendura të "cache" (FA dhe SA), aplikohen rregulla të zëvendësimit sa herë që duhet fshirë ndonjë bllok nga "cache".
- Një politikë optimale e zëvendësimit do të jetë në gjendje të shohë drejt të ardhmes, për të parë se cili bllok nuk do të nevojitet për një kohë më të gjatë
- Edhepse është e pamundur të krijohet një algoritëm plotësisht optimal i zëvendësimit, rekomandohet krahasimi me algoritma ekzistues

Memoria "Cache"

- Rregulla e zëvendësimit varet nga lokaliteti të cilin tentojmë ta optimizojmë – lokaliteti i përkohshëm
- Algoritmi LRU (least recently used) mban evidencë për herën e fundit kur është shfrytëzuar një bllok dhe fshin bllokun i cili ka kohën më të gjatë prej përdorimit
- Mirëpo, LRU është shumë kompleks, sepse duhet t'i qaset "historisë" për çdo bllok, gjë që e ngadalëson "cache"-in

Memoria "Cache"

- First-in, first-out (FIFO) është një politikë e popullarizuar e zëvendësimit të "cache"-it
- Te FIFO, largohet blloku që ka qëndruar më së shumti në "cache", pa marrë parasysh sa është shfrytëzuar
- Rregulla e rastit e zëvendëson një bllok të rastit dhe mund të rezultojë me largimin e ndonjë blloku që përdoret më së shumti

Memoria "Cache"

- Performansa e memories hierarkike matet me kohën efektive të qasjes - effective access time (EAT)
- EAT është një mesatare që merr parasysh "hit rate"
- Në memorien me dy nivele EAT jepet me:

$$EAT = H \times AccessC + (1-H) \times AccessMM$$

H - cache hit rate ; AccessC dhe AccessMM janë kohët e qasjes për "cache" dhe RAM.

Memoria "Cache"

- P.sh. ,memoria qendrore ka kohën e qasjes 200 ns, ndersa "cache" ka kohën e qasjes 10 ns dhe hit rate prej 99%.
- EAT është:

$$0.99(10ns) + 0.01(200ns) = 9.9ns + 2ns = 11ns.$$

- Ky barazim mund të përdoret edhe për nivele të tjera të memories

Memoria "Cache"

- Rregullat e zëvendësimit duhet të kenë kujdes edhe për "blloqet e papastra", të cilat kanë ndërruar përmbajtjen derisa kanë qenë në "cache"
- Këto blloqe duhet të shkruhen në RAM para se të fshihen nga "cache"
- Janë dy lloje të rregullave: "write through" dhe "write back".
- "Write through" bën shkruarjen paralele në "cache" dhe në RAM.

Memoria "Cache"

- "Write back" shkruan në memorie vetëm kur blloku përzgjidhet për fshirje dhe zëvendësim me bllok tjetër
- Mangësi e "write through" është se duhet shkruar në memorie sa herë shkruhet në "cache", gjë që ngadalëson punën. Megjithatë, ky ngadalësim nuk është aq i madh, sepse pjesa më e madhe e qasjeve në "cache" janë lexime e jo shkruarje
- Përparësi e "Write back" është se trafiku në memorie minimizohet, por mangësi është se vlera e bllokut në memorie dhe në "cache" nuk përputhen, gjë që shkakton probleme në sistemet "multitasking".

Memoria virtuale

- Memoria "cache" e shton performansën duke ofruar qasje më të shpejtë memories
- Ndërkaq, memoria virtuale i kontribon kësaj duke shtuar kapacitetin e memories pa kosto shtesë
- Kështu, një pjesë e diskut mund të shërbejë si zgjerim i RAM-it
- Nëse sistemi e përdor faqosjen (paging), memoria virtuale e pjesëton RAM në "faqe" të veçanta që shkruhen (ose faqosen) në disk, nëse nuk nevojiten menjëherë

Memoria virtuale

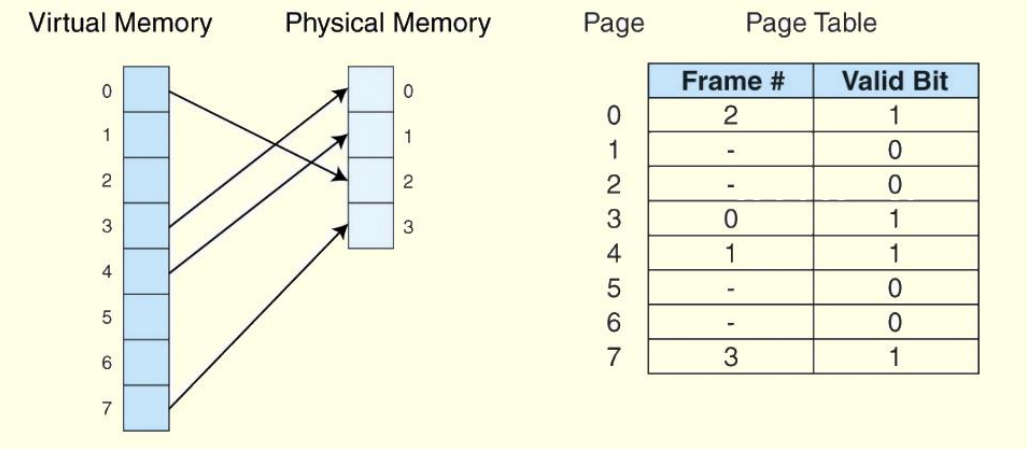
- Adresa fizike është adresa memorike e memories fizike
- Programi i krijon adresat virtuale të cilat pasqyrohen në adresa fizike me anë të menaxhuesit të memories (memory manager)
- Page frames – blloqe në të cilat është e ndarë memoria kryesore
- Pages (faqet)– blloqet në të cilët është e ndarë memoria virtuale
- Page faults paraqiten kur një adrese virtuale (logjike) kërkon që një faqe të lexohet nga disku
- Fragmentimi i memories (memory fragmentation) kur programi nuk e plotëson tërë faqen – krijohen pjesë të vogla të pashfrytëzuara të memories

Memoria virtuale

- Memoria kryesore dhe memoria virtuale janë të ndara në mënyrë të barabartë
- E gjithë hapësira memorike (address space) e një procesi nuk ka nevojë që të jetë në memorie. Vetëm disa pjesë të saj ndodhen në memorien kryesore derisa pjesët tjera ndodhen në disk
- Nuk është e nevojshme që faqet të alokuara në një proces të jenë në menyrë të vazhdueshme (si në disk ashtu edhe në memorie)
- Në këtë mënyrë vetëm faqet e nevojshme janë në memorie në një kohë të caktuar – faqet që momentalisht nuk nevojiten ndodhen në disk

Memoria virtuale

- Informatat për lokacionin e një faqeje – në disk apo në RAM – ruhen në strukturën e të dhënave që quhet tabelë e faqeve - page table
- Një tabelë e faqeve është dhënë më poshtë
- Për çdo proces ekziston nga një tabelë e faqeve



Page	Frame #	Valid Bit
0	2	1
1	-	0
2	-	0
3	0	1
4	1	1
5	-	0
6	-	0
7	3	1

Memoria virtuale

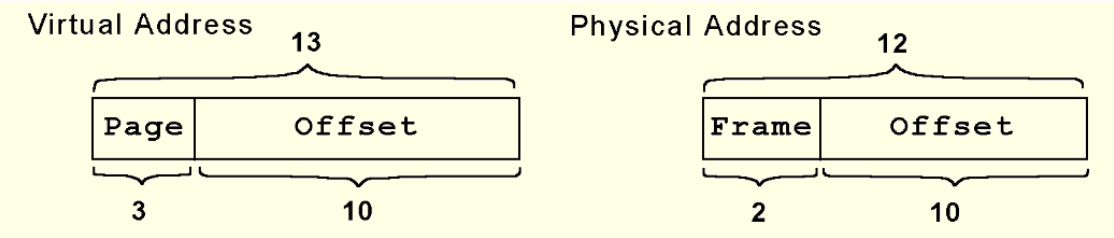
- Kur një proces e gjeneron një adresë të memories virtuale, sistemi operativ (OS) e shndërron këtë në adresë fizike memorike
- Që OS ta realizojë këtë memoria virtuale ndahet në dy fusha: fusha page, dhe fusha offset
  - page përcakton lokacionin e faqes (page), dhe
  - offset përcakton lokacionin në kuader të faqes
- Numri logjik i faqes shndërrohet në numrin fizik të faqes me anë të kërkimit në tabelën e faqeve (page tables)

Memoria virtuale

- Në tabelën e faqeve ruhet edhe biti i validitetit që tregon nëse faqja e caktuar është në memorien kryesore apo në disk
- Nëse CPU dëshiron t'i qaset një faqeje dhe biti i validitetit është 0:
  - Atëherë kemi page fault
  - Nëse është e nevojshme, një faqe largohet nga memoria dhe zëvendësohet me faqen nga disku (dhe i vendoset biti i validitetit në 1)
- Nëse biti i validitetit është 1, numri i faqes virtuale është i zëvendësuar me numrin e faqes fizike
- Qasja në të dhëna bëhet pastaj duke ia shtuar offset numrit të faqes fizike

Memoria virtuale

- P.sh. ta analizojme një sistem qe e ka hapësirën virtuale adresore prej 8KB dhe hapësirën fizike adresore prej 4K Sistemi e shfrytëzon adresimin sipas bajtëve
  - I kemi $2^{13}/2^{10} = 2^3$ faqe virtuale nga 1 KB=2^{10} B
- Një adrese virtuale i ka 13 bit (8K = 2^{13}), ku 3 bit i takojnë fushës page dhe 10 bit i takojnë fushës offset
- Adresa fizike memorike i kërkon 12 bit, dy bitat e parë për numrin e faqes dhe 10 bit për offset



Memoria virtuale

- E kemi tabelën e faqes si më poshtë
- Çka ndodhë kur CPU e gjeneron adresën $5459_{10} = 1010101010011_2$?

		Frame	Valid Bit
Page	0	-	0
	1	3	1
	2	0	1
Page Table	3	-	0
	4	-	0
	5	1	1
	6	2	1
	7	-	0

	Addresses
Page 0 :	0 - 1023
1 :	1024 - 2047
2 :	2048 - 3071
3 :	3072 - 4095
4 :	4096 - 5119
5 :	5120 - 6143
6 :	6144 - 7167
7 :	7168 - 8191

Memoria virtuale

- Adresa 1010101010011 konvertohet në adresën fizike 010101010011 sepse fusha e faqes (page field) 101 është e zëvendësuar me numrin e page frame 01 në anë të tabelës së faqeve

Page Table

	Frame	Valid Bit
Page 0	–	0
1	3	1
2	0	1
3	–	0
4	–	0
5	1	1
6	2	1
7	–	0

	Addresses
Page 0 :	0 - 1023
1 :	1024 - 2047
2 :	2048 - 3071
3 :	3072 - 4095
4 :	4096 - 5119
5 :	5120 - 6143
6 :	6144 - 7167
7 :	7168 - 8191

Memoria virtuale

- Çka ndodhë nese CPU-ja e gjeneron adresen 1000000000100_2?

		Frame	Valid Bit
Page Table	0	–	0
	1	3	1
	2	0	1
	3	–	0
	4	–	0
	5	1	1
	6	2	1
	7	–	0

	Addresses
Page 0 :	0 – 1023
1 :	1024 – 2047
2 :	2048 – 3071
3 :	3072 – 4095
4 :	4096 – 5119
5 :	5120 – 6143
6 :	6144 – 7167
7 :	7168 – 8191

Memoria virtuale

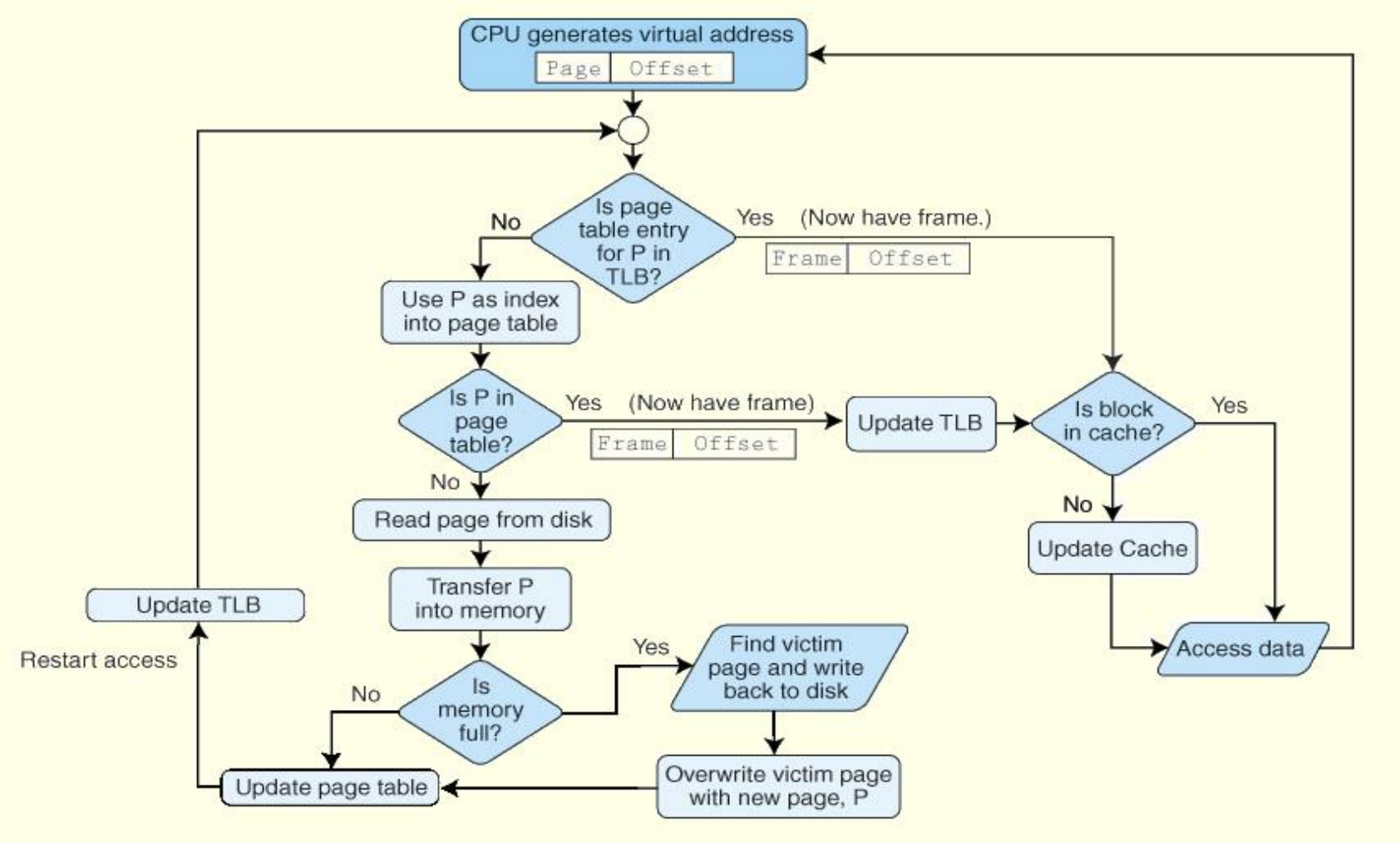
- Më herët kemi thënë se gjatë llogaritjes së EAT merren parasysh të gjitha nivelet e memorieve
- Në këtë llogaritje bën pjesë edhe memoria virtuale dhe duhet ta kemi parasysh edhe kohen e qasjes në tabelën e faqeve
- Nëse memoria kryesore e ka kohen e qasjes 200ns dhe përqindja e page fault është 1% dhe i duhen 10ms të lexojë një faqe prej diskut. Kemi:

$$EAT = 0.99x200ns + 0.01x10ms = 398\ ns$$

Memoria virtuale

- Edhe nëse nuk kemi page faults, EAT do të jetë 400ns sepse memoria është e lexuar dy herë
  - Herën e parë qe t'i qaset tabelës së faqeve
  - Herën e dytë që ta lexojë faqen prej memories
- Pasi që tabelat e faqeve lexohen vazhdimisht është e logjikshme që t'i mbajmë ato në „cache" special të quajtur translation look-aside buffer (TLB)
- TLB janë të organizuar si associative cache dhe e ruajnë pasqyrimin e faqeve virtuale në faqet fizike

Memoria virtuale



Memoria virtuale

- Një metodë tjetër për ta shfrytëzuar memorien virtuale është segmentimi (segmentation)
- Në vend që memoria të ndahet në mënyrë të barabarte në faqe (pages), këtu adresa virtuale është e ndarë në segmente me gjatësi variabile, shpesh nën kontrollin e programuesit
- Segmenti identifikohet përmes shënimeve (entries) në tabelën e segmenteve (segment table)
- Një "shënim" (entry) në tabelën e segmenteve përmban lokacionin memorik të segmentit dhe fundin e lokacionit memorik që në mënyrë indirekte e tregon edhe madhësinë e segmentit
- Në rast të page fault, sistemi operativ e kërkon lokacionin memorik në memorie i cili duhet të jetë mjaft i madh ashtu që ta mbajë segmentin e marrë nga disku

Memoria virtuale

- Faqosja dhe segmentimi shkaktojnë fragmentim
- Faqosja shkakton fragmentimin intern
  - Procesi nuk e mbulon tërë faqen (page)
  - Shumë faqe (pages) mund të kenë fragmente të pashfrytezuara në memorie
- Segmentimi shkakton fragmentimin ekstern.
  - Shfaqet kur hapësira memorike në mes të dy segmenteve është e vogël dhe e pashfrytezueshme

Memoria virtuale

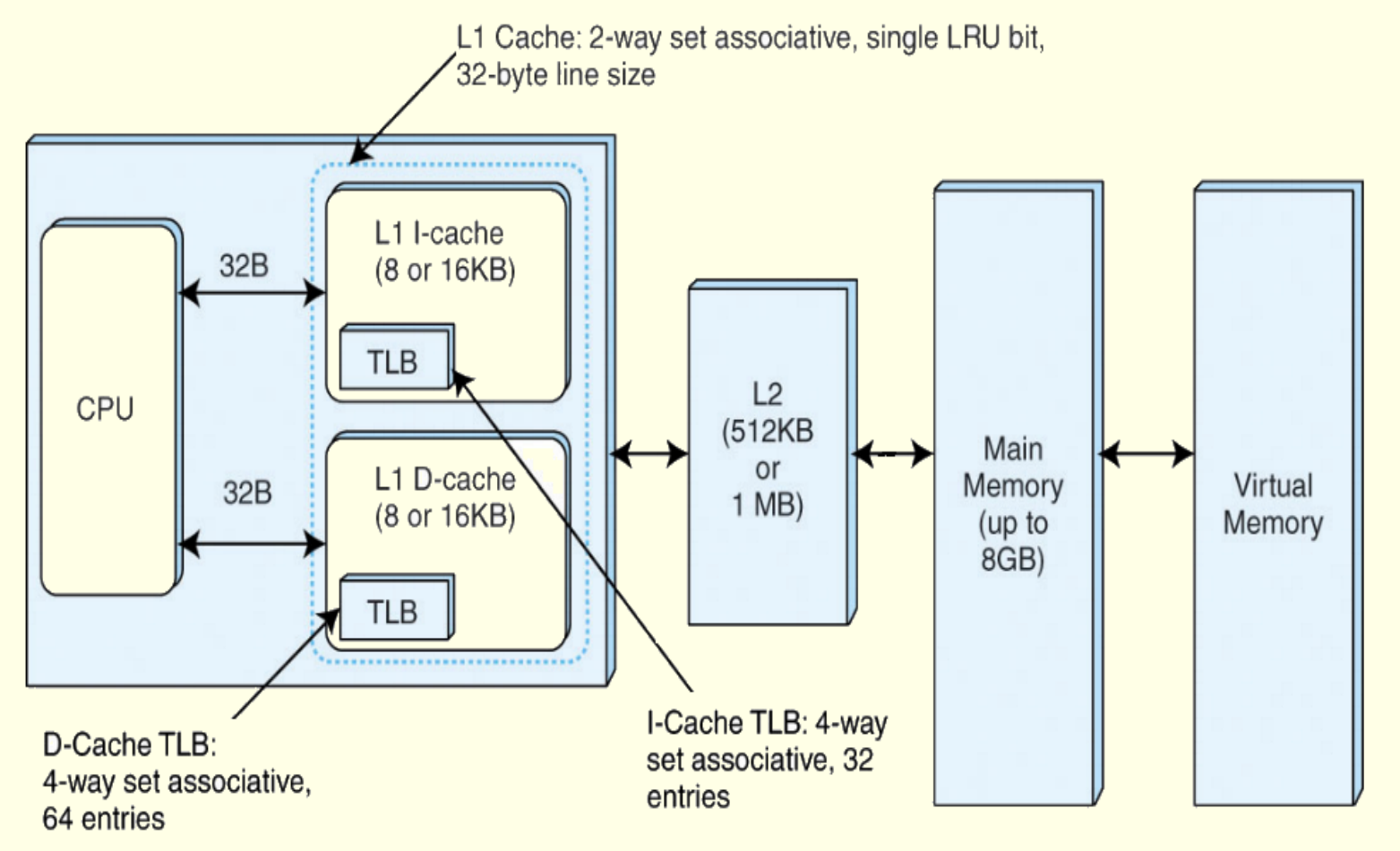
- Faqet (pages) e gjata ngadalësojnë sistemin, nevojitet më shumë hapësirë në memorie për tabelat e faqeve
- Segmentimi mundëson qasje të shpejtë në tabelën e segmenteve, mirepo leximi i segmenteve nga disku është i kushtueshëm
- Faqosja dhe segmentimi mund të kombinohen për t'i shfrytëzuar vetitë pozitive të tyre, duke i ndarë segmentet variabile në faqe me madhësi fikse
- Çdo segment e ka një tabelë të faqes (page table), kjo d.m.th. se adresa memorike do t'i ketë tri fusha, një për segment, një për faqe (page) dhe një për offset

Shembull nga bota reale

- Arkitektura e Pentiumit e përkrahë faqosjen dhe segmentimin dhe ato shfrytezohen në kombinime të ndryshme
- Procesori i shfrytezon dy nivele të "cache"-it (L1 dhe L2), të dyjat kanë madhësinë e bllokut prej 32 B.
- L1 "cache" është pranë procesorit, ndërsa L2 cache gjendet në mes të CPU-së dhe memories
- L1 "cache" përbëhet prej dy pjesëve:
  - instruction cache (I-cache)
  - data cache (D-cache).

Sllajdi vijues paraqet këtë organizim në mënyrë skematike

Shembull nga bota reale



Përmbledhje

- Memoria e kompjutereve organizohet në hierarki
  - me memorien më të shpejte në krye dhe
  - me memorien më të ngadalsheme në fund
- Memoria "cache" ofron qasje të shpejtë në memorien kryesore (RAM), përderisa memoria virtuale e shfrytezon diskun që të krijojë iluzion se kemi më shumë RAM memorie
- "Cache" pasqyron blloqet e RAM në blloqet e cache-it, derisa memoria virtuale pasqyron page frames në faqet virtuale
- Ekzistojne tre lloje të cache-it (direct mapped, fully associative dhe set associative)

Përmbledhje

- Tek fully associative dhe set associative cache, sikurse dhe tek memoria virtuale duhet të përcaktohen rregullat e zëvendësimit
- Rregullat e zëvendësimit janë: LRU, FIFO, ose random
- Memoria virtuale duhet ta zgjidhë problemin e fragmentimit intern tek memoriet e faqosura dhe të fragmentimit ekstern tek memoriet e segmentuara

Pyetje???