Gamze AKSU

171180005

GAZİ ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ

BM311 BİLGİSAYAR MİMARİSİ

M.ALİ AKCAYOL

CACHE COHERENCE PROTOKOLLERİ

Aralık 2020

ÖZET

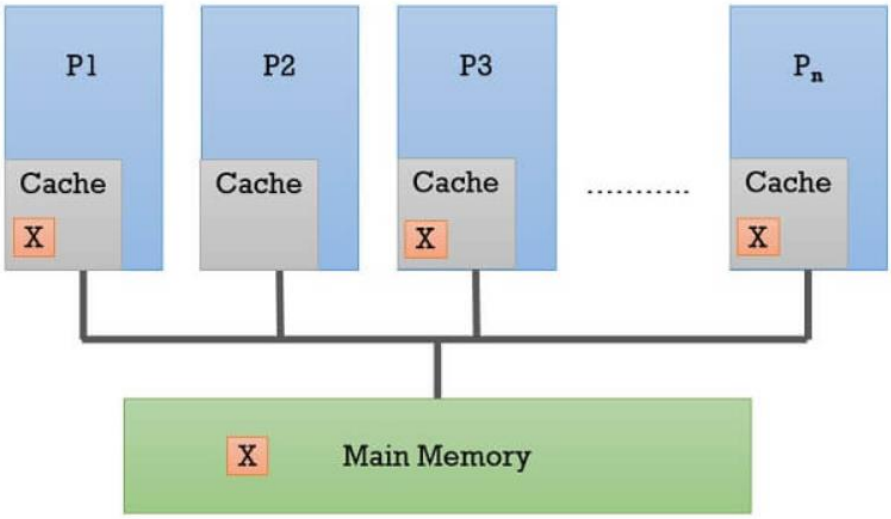
Cache coherence, çok çekirdekli ve çok işlemcili sistemlerde işlemcilerin önbelleklerinde ana belleğin kopyası olan ortak veri bloğu üzerinde işlem yapılırken korunması gereken önbellek tutarlılığına denir. Çok işlemcili sistemlerde cache coherence sağlanmak için birçok protokoller, algoritmalar geliştirilmiştir. Cache coherence için kullanılan protokollerden bazısı şunlardır; MSI, MESI, MOSI, MOESI, MESIF, MERSI, Write-Once, Dragon, Firefly. Bu protokollerin çalışma mantığı, ön bellekte tutulan veri bloklarının üzerinde, yapılmış işlem, yapılacak olan işlem ve farklı şekillerle, blok türlerine göre adlandırılarak sınıflandırılması ile gerçekleşmektedir. Bu araştırma yazısında cache belleklerde veri işleyişi ile birlikte cache coherence tanımı yapılarak, çok çekirdekli ve çok işlemcili sistemlerde cache coherence için kullanılan protokoller anlatılacaktır.

Cache Coherence

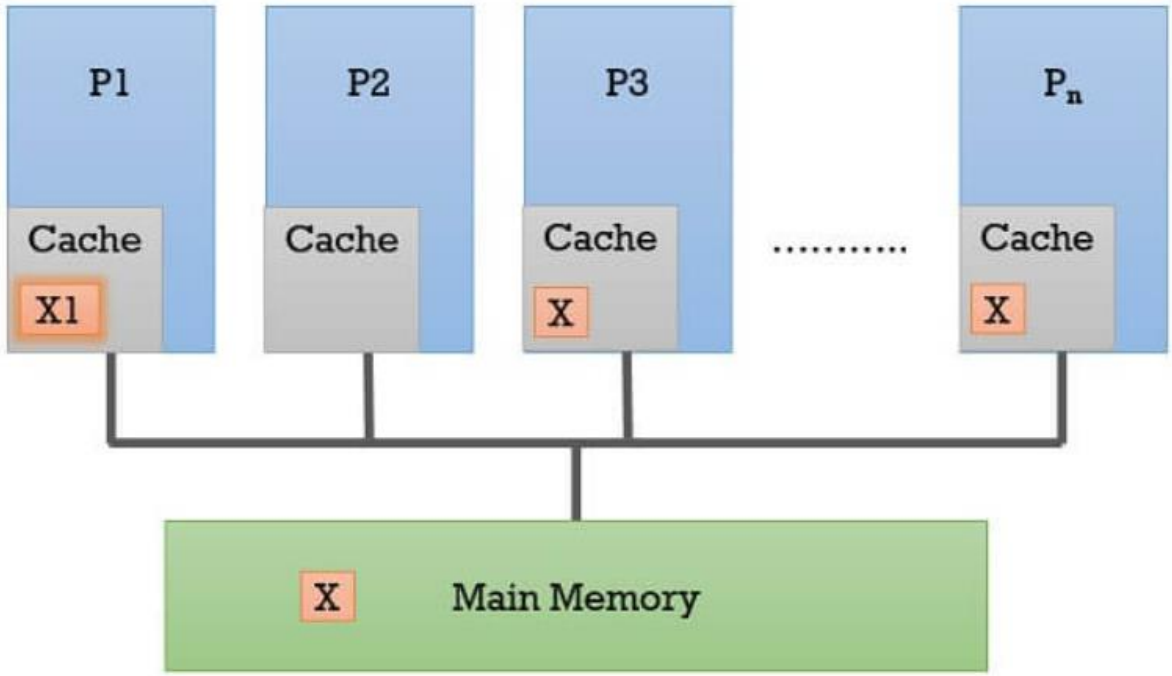
İşlemcilerin veriye ulaşabilmesi için belirli bir süre geçmesi gerekir. Bu süreyi kısaltmanın bazı yolları vardır. Bu yollardan biri de cache belleklerdir. Türkçeye önbellek olarak çevrilirler. Önbellekler işlemciye yakın konumlandırılırlar. Bu sayede veri alış verişi daha hızlı olur. Önbelleklerde depolanan veriler bir önceki hesabın sonucu gibi olabilirken başka bir yerde depolanan verilerin bir kopyası da olabilir. Bu şekilde bir işlemi tekrar yapmaktansa sonuç direk önbelleklerden çekilebilir ya da bir veriyi daha yavaş kaynaklardan okumaktansa hızlıca önbellekten okunabilir. Bu önemli ölçüde performansı arttırır. [1] Bir veri ararken CPU önce önbelleklere orada yoksa da ram'e gider. Eğer orada da yoksa hard disk dediğimiz belleklere gider.[2]

Hafıza birimlerine CPU tarafından erişim süresi kısaldıkça bit başına maliyeti artar. Aynı şekilde hafıza birimlerinin kapasitesi arttıkça erişim süresi artar. Bu yüzden önbelleklerin kapasitesi azdır ama ram'lere göre daha hızlıdır. Şu anda kullandığımız bilgisayarlarda sahip olduğumuz teknolojideki en son hızı kullanamayız. Yani bilgisayarlarımızın tüm hafızası ön belleklerden oluşmaz. Çünkü eğer öyle olsaydı bu çok maliyetli olurdu. Bir optimizasyon işlemine tabi tutularak en optimum dengede bu bellekler parçalanmaktadır. [1][3]

Günümüzde çok çekirdekli işlemciler ve çok işlemcili sistemlerle önbellek yapısı birlikte kullanılır. Bu şekilde bir işlemcinin erişip değiştirdiği bir veriye diğer işlemcinin de erişme ihtimali olabilir. Yani yanlış veriye ulaşabilir. Ortak bir bellek paylaşıldığı zaman bir verinin değiştiğini diğer işlemcilere de haber vermek gerekir. Bu hatalı veya yanlış sonuçların çıkmasını engeller. Bu şekilde Cache Coherence (önbellek tutarlılığı) sağlanmış olur. [4][5]



Yukarıda çok işlemcili bir ortamda main memory, işlemciler ve önbellekleri görülmektedir. Tüm işlemciler için ortak bir önbellek tutmak işlemleri yavaşlatacaktır. Çünkü her işlemciye ortak bir önbellek için önbelleğin kapasitesinin arttırılması gerekir. Bu da performansı kötü etkiler. Performansı arttırmak için her işlemcinin kendi önbelleği olur. Yukarıdaki şekilde de görüldüğü üzere X bir veri bloğu ve hem ana bellekte hem de bazı önbelleklerde kopyası bulunmaktadır. Bu X veri bloklarından bir değişirse diğerlerinin de değişmesi gereklidir. Değişmediği zaman cache coherence problemleri ortaya çıkar. [6]



Tutarlılık sorununun üç temel nedeni vardır. Bunlar:

1- Yazılabilir verilerin paylaşılması(Sharing of writable data.)

2- İşlem geçişi (Process migration)

3- Giriş Çıkış olayları nedeniyle tutarsızlık(Inconsistency due to I/O.) [7]

Cache coherence problemini çözmek için aşağıdaki protokoller tasarlanmıştır.

- **MSI Protokolü**

Çok işlemcili sistemlerde kullanılan temel bir cache coherence protokolüdür. MSI protokolünde her önbellek bloğu 3 olası duruma sahiptir.

- **M**odified(Değiştirilmiş): Veri bloğu ön bellekte değiştirilmiştir. Önbellekteki veri ana bellekteki veriyle uyumlu değildir. Bir veri bloğu "M" durumunda olan bir önbellekte, bu veri bloğu çıkartılacağı zaman onu ana belleğe yazmakla sorumludur.
- **S**hared(Paylaşılan): Bu durumda veri bloğu değiştirilmemiştir ve en az bir başka önbellekte de kopyası vardır. Bu veri bloğu çıkartılacağı zamana ana belleğe yazmaya gerek yoktur.
- **I**nvalid(Geçersiz): Bu blok ya geçerli önbellekte mevcut değildir ya da bir veri yolu talebi ile geçersiz kılınmıştır ve eğer blok bu önbellekte saklanacaksa, bellekten veya başka bir önbellekten getirilmelidir.

Bu durumlar, önbellekler ve main memory arasındaki iletişim ile devam ettirilir. Veri blokları okunduğunda veya yazıldıklarında ya da başka bir önbellekte olan okuma yazma olaylarından haber aldığında önbelleklerin farklı sorumlulukları vardır. “M” veya “S” durumlarında bloklara sahip bir önbelleğe okuma talebi geldiğinde önbellek verileri sağlar. Blok önbellekte değilse (“I” durumunda), bloğun başka bir önbellekte “M” durumunda olup olmadığına bakması ve “M” durumunda olmadığını doğrulaması gerekir. “M” durumundaki bir blok için önbelleğe yazma isteği ulaştığında, önbellek verileri yerel olarak değiştirir. Blok “S” durumundaysa, önbellek “S” durumunda bloğa sahip olabilecek başka önbellekleri bloğu çıkarmaları gerektiği konusunda bilgilendirmelidir. Daha sonra veriler yerel olarak değiştirilebilir. Blok “I” durumundaysa, önbellek “S” veya “M” de bloğu içerebilecek diğer önbellekleri, bloğu boşaltmaları gerektiğini bildirmelidir. [8]

Herhangi bir önbellek çifti için, belirli bir önbellek satırının izin verilen durumları aşağıdaki gibidir:

	M	S	I
M	✗	✗	✓
S	✗	✓	✓
I	✓	✓	✓

İki önbellek arasındaki erişebilirlik bu şekilde gösterilebilir. Örneğin “S” durumundaki bir bloğa sahip önbelleğe diğer önbelleğin “M” durumundaki blokları erişemezken “S” ve “I” durumundaki bloklar erişebilir.

- **MESI protokolü**

MESI protokolü bir invalidate-based(geçersiz kılma tabanlı) bir cache coherence protokolüdür. Illionis Üniversitesinde geliştirilmesi nedeniyle Illionis protokolü olarak da geçmektedir. (Write-back caches) geri-yazma önbellekleri, genellikle (write-through caches) içe yazma önbelleklerinin boşa harcadığı bant genişliğinden tasarruf sağlayabilirler. Geri-yazma önbelleklerindeki veriler ana bellektekinden farklıdır. Bir blok başka bir önbellekte bulunuyorsa, bir hata durumunda MESI protokolü önbellekten önbelleğe bir transfer gerektirir. Bu protokol MSI protokolüne göre daha az işlem olmasını sağlar ve bu da performansı olumlu yönde etkiler. [9]

- **M**odified: Bu önbellek satırı yalnızca geçerli önbellekte bulunur ve bulunduğu değer ana bellekteki değerden farklıdır. Artık geçerli olmayan ana bellek durumunun başka her hangi bir şekilde okunmasına izin vermeden önce ana belleğe geri yazması gereklidir. Geri-yazma olayı durumu Shared(paylaşılan) olarak değiştirir.
- **E**xclusive(Ayrıcalıklı): Bu önbellek satırı da modified (değiştirilmiş) de olduğu gibi yalnızca geçerli önbellekte bulunur ama bulunduğu değer ana bellekteki ile aynıdır. Okuma talebine yanıt olarak herhangi bir zamanda Shared (Paylaşılan) duruma değiştirilebilir. Alternatif olarak, yazılırken Modified (Değiştirilmiş) durumuna da geçiş yapabilir.
- **S**hared(Paylaşılan): Bu önbellekteki satırın diğer önbelleklerde de bir kopyasının olabileceğini gösterir. Buradaki değer de ana bellekle aynıdır. Herhangi bir zamanda Invalid (Geçersiz) durumuna dönüşebilir.
- **I**nvalid (Geçersiz): Bu satırın kullanılmamış olduğunu gösterir.

Bir blok "M" veya "E" durumundaysa bloğun diğer önbelleklerdeki kopyaları "I" olarak işaretlenir. Herhangi bir önbellek çifti için, önbellek satırının izin verilen durumları şöyle gösterilebilir. [9]

	M	E	S	I
M	✗	✗	✗	✓
E	✗	✗	✗	✓
S	✗	✗	✓	✓
I	✓	✓	✓	✓

- **MOSI protokolü**

MOSI protokolü MSI Protokolünün geliştirilmiş halidir. MSI de bulunan durumlara ek olarak Owned(Sahipli) durumu vardır.

- **O**wned(Sahipli): Mevcut işlemcinin bu bloğa sahip olduğunu ve blok için diğer işlemcilerden gelen taleplere hizmet vereceğini belirtir. Birçok önbellek bir bloğun en son ve doğru değerini tutabilir. Ana bellekteki değer de doğru olsun ya da olmasın bir seferde yalnızca bir önbellek bir blok için Owned(Sahipli) durumunda olabilir. Aynı bloğun kopyalarına sahip diğer önbelleklerde bu blok Shared(Paylaşılan) durumunda olmalıdır. [10]

	M	O	S	I
M	✗	✗	✗	✓
O	✗	✗	✓	✓
S	✗	✓	✓	✓
I	✓	✓	✓	✓

Diğer kalan 3 durum da MSI ile aynıdır. Bunlar, Modified(Değiştirilmiş), Shared(Paylaşılan) ve Invalid(Geçersiz) durumlarıdır.

Herhangi bir önbellek çifti için, önbellek satırının izin verilen durumları yandaki gibi gösterilebilir.

- **MOESI protokolü**

	M	O	E	S	I
M	✗	✗	✗	✗	✓
O	✗	✗	✗	✓	✓
E	✗	✗	✗	✗	✓
S	✗	✓	✗	✓	✓
I	✓	✓	✓	✓	✓

MOESI protokolünde MSI protokolüne ek olarak Exclusive(Ayrıcalıklı) ve Owned(Sahipli) durumları vardır. Bu durumlarla beraber 5 duruma sahiptir. Yaygın olarak kullanılan protokollerin tüm olası durumlarına sahiptir. MOSI ve MESI protokollerinin birleşimi gibi düşünülebilir. Ek durumlardaki Exclusive MESI protokolünde, Owned durumu ise MOSI protokolünde açıklanmıştır. Herhangi bir önbellek çifti için, önbellek satırının izin verilen durumları şöyle gösterilebilir. [11]

- **MESIF Protokolü**

MESIF protokolü Intel firması tarafından geliştirilen bir cache coherence protokolüdür. Bu protokol de MOESI de olduğu gibi bir blok için 5 duruma sahiptir. Bunlar Modified(Değiştirilmiş), Exclusive(Ayrıcalıklı), Shared(Payllaşılan), Invalid(Geçersiz) ve Forward(İleri) durumlarıdır. [12]

Modified, Exclusive, Shared ve Invalid durumları MESI protokolü ile aynı işlevlere sahiptir.

- Forward(İleri): Shared(Paylaşılan) durumunun özel bir biçimidir. Forward durumu bir önbelleğin verilen bir bloğun herhangi bir isteği için yanıtlayıcı olarak atandığını gösterir. Herhangi bir önbellek Shared(Paylaşılan) durumundaki bir bloğu tutuyorsa en fazla bir başka önbelleğin bunu Forward durumunda tutmasını sağlar.

Önbellek "S" veya "F" durumlarındaki bir bloğu tek taraflı olarak geçersiz kılabileceğinden "S" durumundan kopyalar olsa bile hiçbir önbelleğin "F" durumunda bir kopyasına sahip olmaması mümkündür. Bu durumda ana bellekten bu blok için gelen istek karşılanır. Bu daha az verimlidir ama yine de doğrudur. MESI protokolünden temel farkı, okuma için önbellek satırının bir kopyasına yönelik bir isteğin, önbelleğe her zaman "F" durumunda girmesidir. "S" durumuna girmenin tek yolu, ana bellekten bir okuma talebini karşılamaktır. [12]

	M	E	S	I	F
M	✗	✗	✗	✓	✗
E	✗	✗	✗	✓	✗
S	✗	✗	✓	✓	✓
I	✓	✓	✓	✓	✓
F	✗	✗	✓	✓	✗

Herhangi bir önbellek çifti için, önbellek satırının izin verilen durumları solda gösterilmiştir. Gereksiz yanıtları bastırırken paylaşılan önbelleklerden okuma isteklerini karşılamaya yönelik başka teknikler de vardır, ancak yalnızca tek bir belirlenmiş önbellek yanıtına sahip olmak, Exclusive(Ayrıcalıklı) duruma geçiş için gerektiğinde tüm kopyaların geçersiz kılınmasını kolaylaştırır. [12]

- **MERSI protokolü**

MERSI protokolü PowerPC G4 tarafından kullanılan bir cache coherence protokolüdür. Bu protokol bir blok için 5 duruma sahiptir. Bunlar Modified (Değiştirilmiş), Exclusive (Ayrıcalıklı), Read Only or Recent (SaltOkunur), Shared (Paylaşılan) ve Invalid (Geçersiz) durumlarıdır. Modified, Exclusive, Shared ve Invalid durumları MESI protokolü ile aynı işlevlere sahiptir. [13]

- **R**ead-Only or Recent: Exclusive(Ayrıcalıklı) durumuna benzerdir. Verilerin tek temiz ve geçerli bir kopyası vardır. “E” durumunun tersi olarak işlemcinin önbellek bloğunu değiştirip “M” durumuna geçebilmesi için önce “R” durumunda olması gerekir.

Herhangi bir önbellek çifti için, önbellek satırının izin verilen durumları şöyle gösterilebilir.

	M	E	R	S	I
M	✗	✗	✗	✗	✓
E	✗	✗	✗	✗	✓
R	✗	✗	✗	✗	✓
S	✗	✗	✗	✓	✓
I	✓	✓	✓	✓	✓

- **Write-Once protokolü**

Write-Once protokolü bilgisayar belleğine ardışık yazma işlemlerinde genel veri yolu trafiğini azaltmayı amaçlar. İlk yazma işleminde write-through ve sonraki tüm yazma işlemlerinde write-back işlemlerinin optimizasyonudur. [14] James Goodman, bu protokolü ilk olarak "Using Cache Memory to Reduce Processor-Memory Traffic"("İşlemci-Bellek Trafiğini Azaltmak için Önbellek Belleğini Kullanma") bölümünde 10. Uluslararası Bilgisayar Mimarisi Sempozyumunda açıklanmıştır. (ISCA 1983). [15]Bir önbellek satırı 4 farklı durumda olabilir.

- **I**nvalid(Geçersiz): Önbellek bloğu önbellekte yoktur ya da blokta belleğin tutarsız bir kopyası vardır. Yani bloktaki veri ile ana bellekteki veri uyuşmaz.

- **V**alid(Geçerli): Bu blok, belleğin tutarlı bir kopyasına sahiptir. Yani bloktaki veriler ile ana bellekteki veriler uyuşur. Bu veriler muhtemelen paylaşılabilir, ancak değiştirilmezler.
- **R**eserved(Ayrılmış): Blok, belleğin tek kopyasıdır, ancak yine de tutarlıdır. Blok değiştirilirse geri yazma gerekmez.
- **D**irty(Kirli): Blok belleğin tek kopyasıdır ve tutarsızdır. Yani ana bellek ile uyumsuzdur. Sadece bu durumda önbellekte blok değiştirildiğinde write-back olur.

Herhangi bir önbellek çifti için, önbellek satırının izin verilen durumları şöyle gösterilebilir.

	I	V	R	D
I	✓	✓	✓	✓
V	✓	✓	✗	✗
R	✓	✗	✗	✗
D	✓	✗	✗	✗

- **Dragon Protokolü**

Dragon Protokolü bir güncelleme tabanlı cache coherence protokolüdür. Çok işlemcili sistemlerde kullanılırlar. Birden çok işlemcide önbelleğe alınan tüm değerlerin doğrudan güncellenmesiyle yazma yayılımı (write propagation) uygulanır. Önbelleğe bir yazmanın ardından diğer işlemciler tarafından birkaç okuma yapıldığında verimli bir şekilde çalışır. Çünkü güncellenmiş önbellek bloğu tüm işlemcilerle ilişkili önbelleklerde kolaylıkla kullanılabilir. Geçersiz kılma protokollerinden farklı olarak, bloğun ana bellekte değil, yalnızca işlemcide güncel olması gerekir. Önbellek bloğu kaldırıldığında ana belleği güncellemek işlemcinin sorumluluğundadır. Dragon protokolünde her blok için sahip olabileceği 4 durum vardır. [16]

- **E**xclusive-clean: Bu durum şunu ifade eder: Bu blok geçerli işlemci tarafından ilk olarak getirilmiştir ve o zamandan beri başka bir işlemci tarafından erişilmemiştir.
- **S**hared **c**lean: Önbellek bloğu kesinlikle farklı işlemcilerin önbelleğinde de vardır ve bloğu yazan son işlemcinin geçerli işlemci değildir.

- **S**hared **m**odified: Bloğun birden fazla işlemcinin önbelleklerinde bir kopyası bulunur ve geçerli işlemcinin bloğu değiştiren son işlemci olduğunu gösterir. Sonuç olarak, geçerli işlemci bloğun sahibi olarak adlandırılır.
- **M**odify: Bu bellek bloğuna yalnızca bir işlemcinin sahip olduğunu gösterir. Hafızadan getirildiğinden beri değerinin değişmiştir.

Herhangi bir önbellek çifti için, önbellek satırının izin verilen durumları şöyle gösterilebilir.

	E	Sc	Sm	M
E	✗	✗	✗	✗
Sc	✗	✓	✓	✗
Sm	✗	✓	✗	✗
M	✗	✗	✗	✗

- **Firefly Protokolü**

DEC Firefly'da kullanılmıştır. DEC Systems Research Center tarafından geliştirilmiştir. 3 duruma sahip bir yazma güncelleme cache coherence protokolüdür. Dragon protokolünden farklı olarak, ana belleği günceller. Bu nedenle Dragon protokolündeki Shared clean ve Shared modified durumları, firefly protokolünde birbirinden ayırt edilemezler. Firefly protokolünde her bir blok için 3 durum vardır. [17]

- Valid-Exclusive(Geçerli-Ayrıcalıklı): Bu önbellek bloğu sadece bir önbellekte bulunur. Blok geçerli ve temizdir. Yani bloktaki veri ile ana bellekteki veri birbiriyle eşleşir.
- Shared(Paylaşılan): Blok geçerli ve temizdir. Yani bloktaki veri ile ana bellekteki veri birbiriyle eşleşir. Valid-Exclusive durumundan farklı olarak bu blok birden çok önbellekte bulunabilir.
- Dirty(Kirli): Blok ana belleğin önbelleklerdeki tek kopyasıdır. Kirlidir, yani bloktaki veri değiştirilmiştir. Ana bellekteki veriyle uyuşmaz. Bu blok önbellekte değiştirildiğine göre write-back gerektirir.

KAYNAKÇA

1. https://en.wikipedia.org/wiki/Cache_(computing)
2. https://www.chip.com.tr/blog/enginsaklidarkexecut/cache-bellek-nedir-ne-ise-yarar-nasil-calisir_4455.html
3. http://www.mademir.com/2010/10/cache-cohorancy.html
4. https://en.wikipedia.org/wiki/Cache_coherence
5. http://www.mehmetduran.com/Blog/Makale.html/Cache-Coherence-Kavrami/194
6. https://binaryterms.com/cache-coherence.html
7. https://www.geeksforgeeks.org/cache-coherence-protocols-in-multiprocessor-system/
8. https://en.wikipedia.org/wiki/MSI_protocol
9. https://en.wikipedia.org/wiki/MESI_protocol
10. https://en.wikipedia.org/wiki/MOSI_protocol
11. https://en.wikipedia.org/wiki/MOESI_protocol
12. https://en.wikipedia.org/wiki/MESIF_protocol
13. https://en.wikipedia.org/wiki/MERSI_protocol
14. https://en.wikipedia.org/wiki/Write-once_(cache_coherence)
15. https://www.scss.tcd.ie/Jeremy.Jones/vivio%205.1/caches/writeOnceHelp.htm
16. https://en.wikipedia.org/wiki/Dragon_protocol
17. https://en.wikipedia.org/wiki/Firefly_(cache_coherence_protocol)