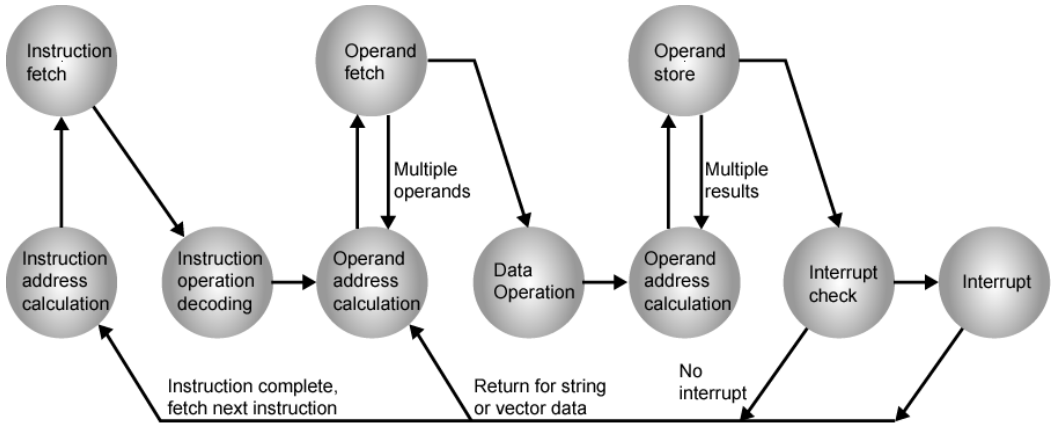
1-) Instruction Cycle State Diagram'ı çizip herbir state için gerçekleştirilen işlemleri detaylı bir şekilde açıklayınız.

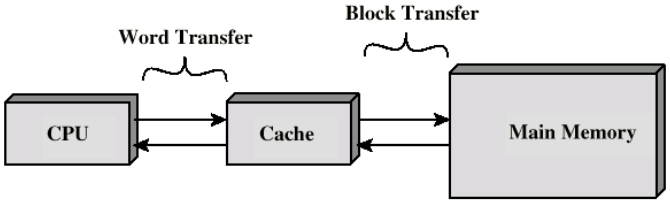


Instruction state cycle'da üstteki kısımlar CPU dışında alttaki kısımlar CPU içinde gerçekleştirilir. Vektör ve string işlemlerinde işlem yapan komut değişmediği için operand address calculation durumu sürekli tekrarlanır. Interrupt check durumunda kesme gelmemeişse sonraki işleme geçilir. Sonraki işlem yeni bir komut alınması veya operand alınması olabilir. Operand fetch ve operand store durumları birden fazla operand için tekrarlanır.

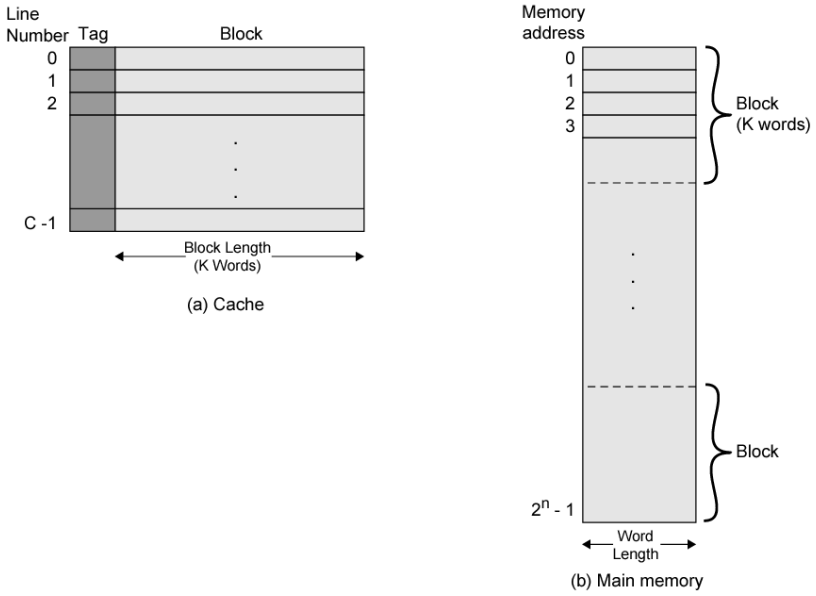
Instruction address calculation: Sonraki komutun adresi hesaplanır.
Instruction fetch: Adresi hesaplanan komut alınır.
Instruction operation decoding: Komutun ne iş yapacağı kontrol birimi tarafından belirlenir.
Operand address calculation: Adresleme modları kullanılarak işlem için gerekli operandların adresleri hesaplanır.
Operand fetch: Adresi hesaplanan operandlar alınır.
Data Operation: Komut ve operandlar kullanılarak gerekli işlem ALU tarafından yapılır.
Operand address calculation: Komut sonuç üretmişse (ADD, SUB, …) yazılacağı adres hesaplanır.
Operand store: Adresi hesaplanan sonuç operand kaydedilir.
Interrupt check: Kesme gelip gelmediği kontrol edilir.
Interrupt: Kesme gelmişse ilgili subroutine çalıştırılır.

2-) Cache belleklerin yapısı ve kullanım amacı hakkında bilgi veriniz. Mapping function'lar hakkında detaylı bilgi veriniz.

Önbellek kullanımının temel amacı, sık kullanılan bilgileri CPU'ya daha yakın ve hafızadan çok daha hızlı önbellekte saklayarak performansı artırmaktır. Önbellek, main memory ile CPU arasına yerleştirilir. CPU önce önbelleğe erişir ve aradığı veri yoksa main memory'ye erişir. Eğer aranan veri main memory'de ise içinde bulunduğu blok ile birlikte alınır. Önbelleğe ve CPU'ya aktarılır. Hafızadan önbelleğe blok transfer, önbellekten CPU'ya word transfer edilir.

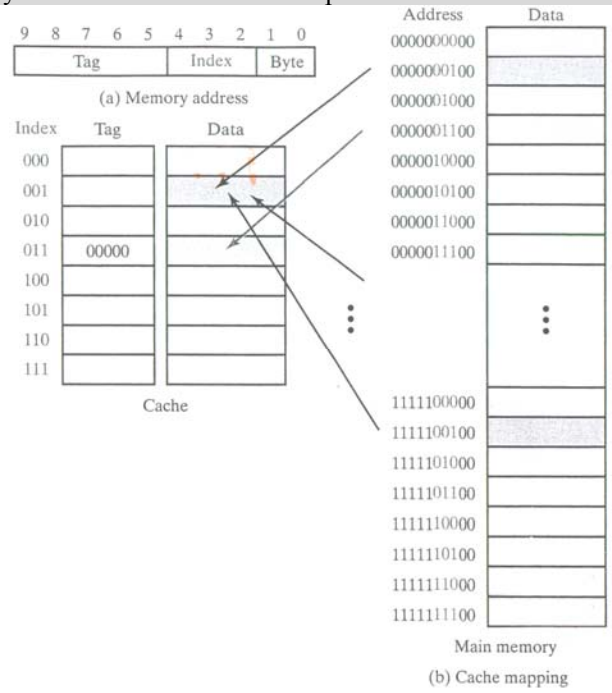


Önbellek ve main memory yapısı aşağıdaki şekilde görülmektedir. Şekilde M blok sayısıdır. $M = 2^n / K$ ve $C << M$ 'dir. Hafızadan sık kullanılan veya yakın zamanda sık kullanılacak bilgilerin belirlenerek önbelleğe aktarılması için farklı yaklaşımlar kullanılır. Önbellekteki satır sayısı main memory'den çok az olduğu için eşleştirme fonksiyonu kullanılarak aktarma yapılır. Mapping function hafızadaki bir bloğun önbelleğe nasıl yerleştirileceğini belirler. Direct, associative ve set associative olarak üç teknik kullanılır.



Direct mapping

Şekilde önbellek satır boyutu 32-bit alınmıştır. Hafızada her satır 4-byte (1 blok) kapasitedir. Önbellek adresleme 3-bit ile yapılmaktadır. Hafızadaki 256 blok önbellekteki 8 satıra eşleştirilmektedir. Eşleştirme modüler aritmetiğe göre yapılır.

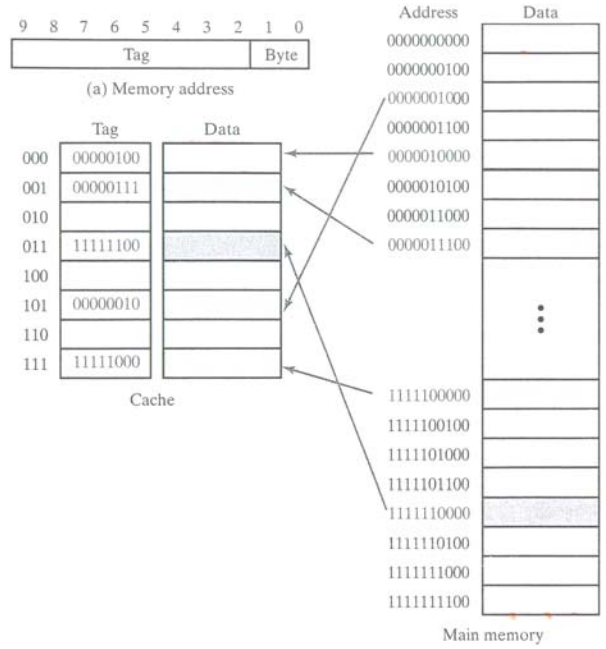


Avantaj / dezavantaj

Oluşturmak basit ve ucuzdur. Bir blok sadece bir satıra yazılabilir. Aynı satıra eşleşen iki blok sürekli çalıştığında performans düşer.

Associative mapping

Direct mapping'teki bir bloğun sadece satıra eşleştirilmesi dezavantajı ortadan kaldırılır. Bir blok önbellekte istenilen satıra eşleştirilir. Aranan bilginin önbellekte olup olmadığı tüm satırlarda eşzamanlı kontrol edilir. Hafızadan alınan bloğun önbellekte hangi satıra yazılacağı replacement algoritmasıyla belirlenir.

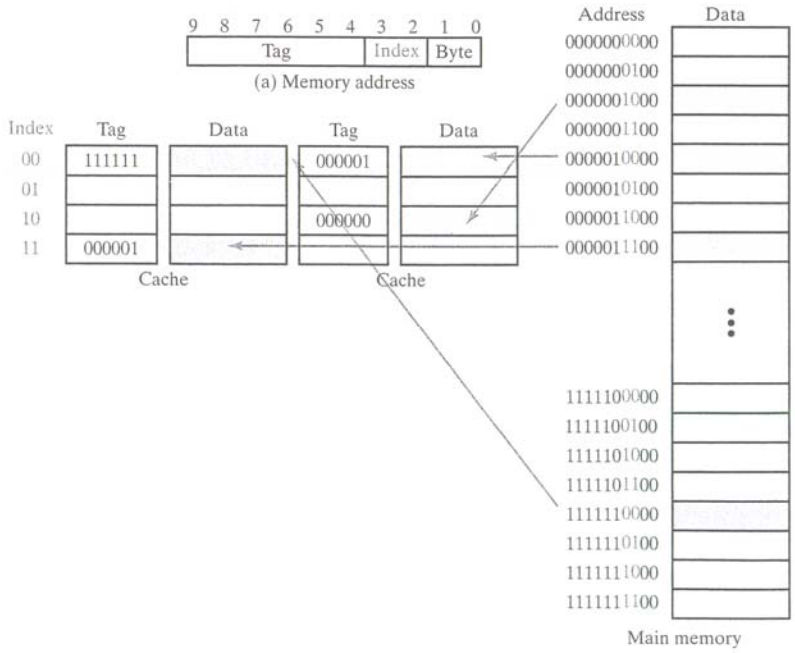


Avantaj / dezavantaj

Yapısı karmaşıktır. Bir blok uygun olan bir satıra yazılabilir. Önbellekte eşzamanlı arama hızı düşüktür.

Set associative mapping

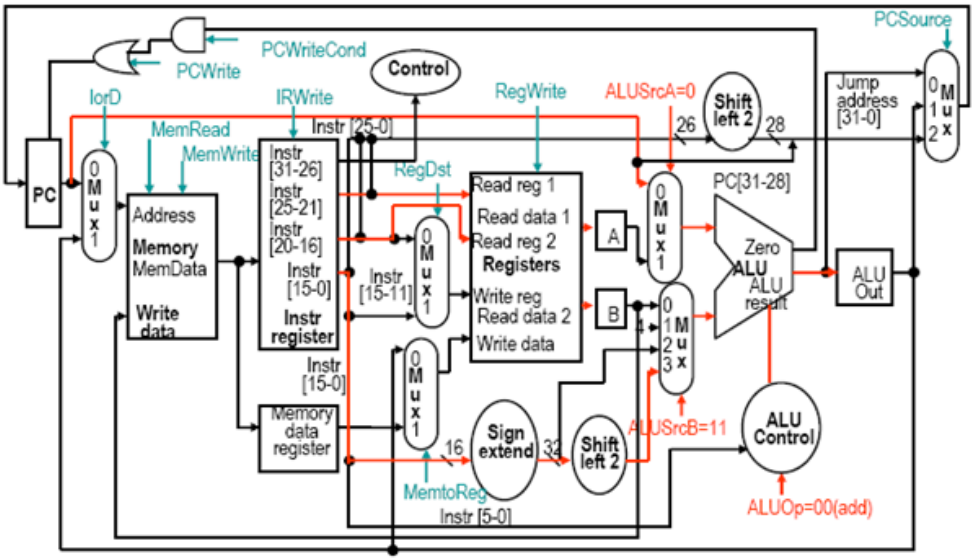
Direct mapping ve associative mapping birleşimidir. Önbellek belirli sayıda kümeden oluşur. Her küme kendi içinde associative yapıdadır ve hafızadan alınan bloğun kümede hangi satıra yazılacağı replacement algoritmasıyla belirlenir.

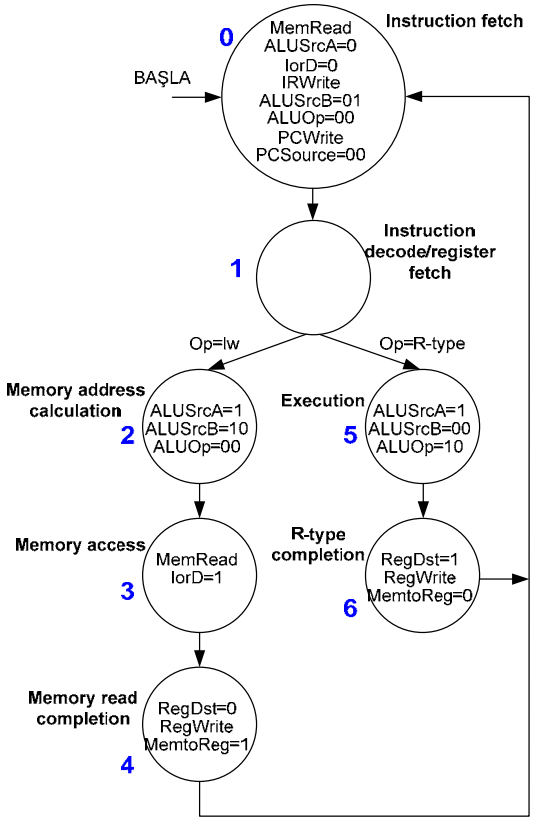


Avantaj / dezavantaj

Yapısı direct mapping'e göre karmaşıktır. Bir blok sadece kendisine ait bir kümedeki istenilen satıra yazılabilir. Önbellekte küme içinde eşzamanlı arama hızı düşüktür.

3-) Aşağıdaki veriyolunda multicycle olarak **load** ve **R-type** komutların çalışmasını sonlu durum makinesiyle göstererek anlatınız.





0 ve 1 durumları tüm komutlar için aynıdır. Instruction fetch durumunda MemRead ile hafızadan komut okuma işlemi etkin yapılır. ALUSrcA = 0 ve ALUSrcB = 01 yapılarak PC değerine 4 eklenir. IRWrite ile okunan komut IR 'a yazılır. IorD = 0 yapılarak PC değeri hafızaya gönderilir. PCSource = 00 yapılarak 4 eklenen PC değeri tekrar PC register'ına aktarılır. PCWrite etkin yapılarak PC değerini şartsız değiştirir.

Instruction decode/register fetch durumunda IR'deki komutun ilgili parçaları RegFile ve mux'lara aktarılarak decode edilir. Herhangi bir kontrol işaretine gerek yoktur.

Eğer gelen komut lw ise 2.duruma, R-type ise 5.duruma geçilir.

Gelen komut lw ise ALUSrcA=1 ve ALUSrcB=00 yapılarak PC ile sign extend yapılan adres ALU'ya giriş olarak verilir. ALUOp = 00 yapılarak toplama işlemi yapılır. Memory access durumunda MemRead ile hafızadan okuma etkin yapılır ve IorD = 1 ile okunacak adres ALU çıkışında hesaplanan adres olarak alınır. Okunan değer Memory data register'a kaydedilir. Memory read completion durumunda lw komutunun sonlandırılması yapılır. RegDst = 0 ile yazılacak register seçilir, MemtoReg = 1 ile MDR çıkışı RegFile'a aktarılır. RegFile'a yazma işlemi etkin yapmak için RegWrite etkin yapılır.

Gelen komut R-type ise ALUSrcA=1 ve ALUSrcB=10 yapılarak RegFile'dan okunup A ve B register'larına kaydedilen değerler ALU'ya giriş olarak verilir. ALUOp=10 yapılarak instruction içindeki function field ile gerekli işlemin seçilmesi sağlanır. RegDst=1 yapılarak instruction içindeki rd alanı hedef register'ı seçer. MemtoReg = 0 yapılarak ALU çıkışı RegFile'a aktarılır. RegFile'a yazma işlemi etkin yapmak için RegWrite etkin yapılır.