
# Yıldız Teknik Üniversitesi,
# Bilgisayar Mühendisliği Bölümü,
# 2021-2022 Öğretim Yılı Bahar yy.,
# BLM2022 Bilgisayar Donanımı,
# Ödev – 2

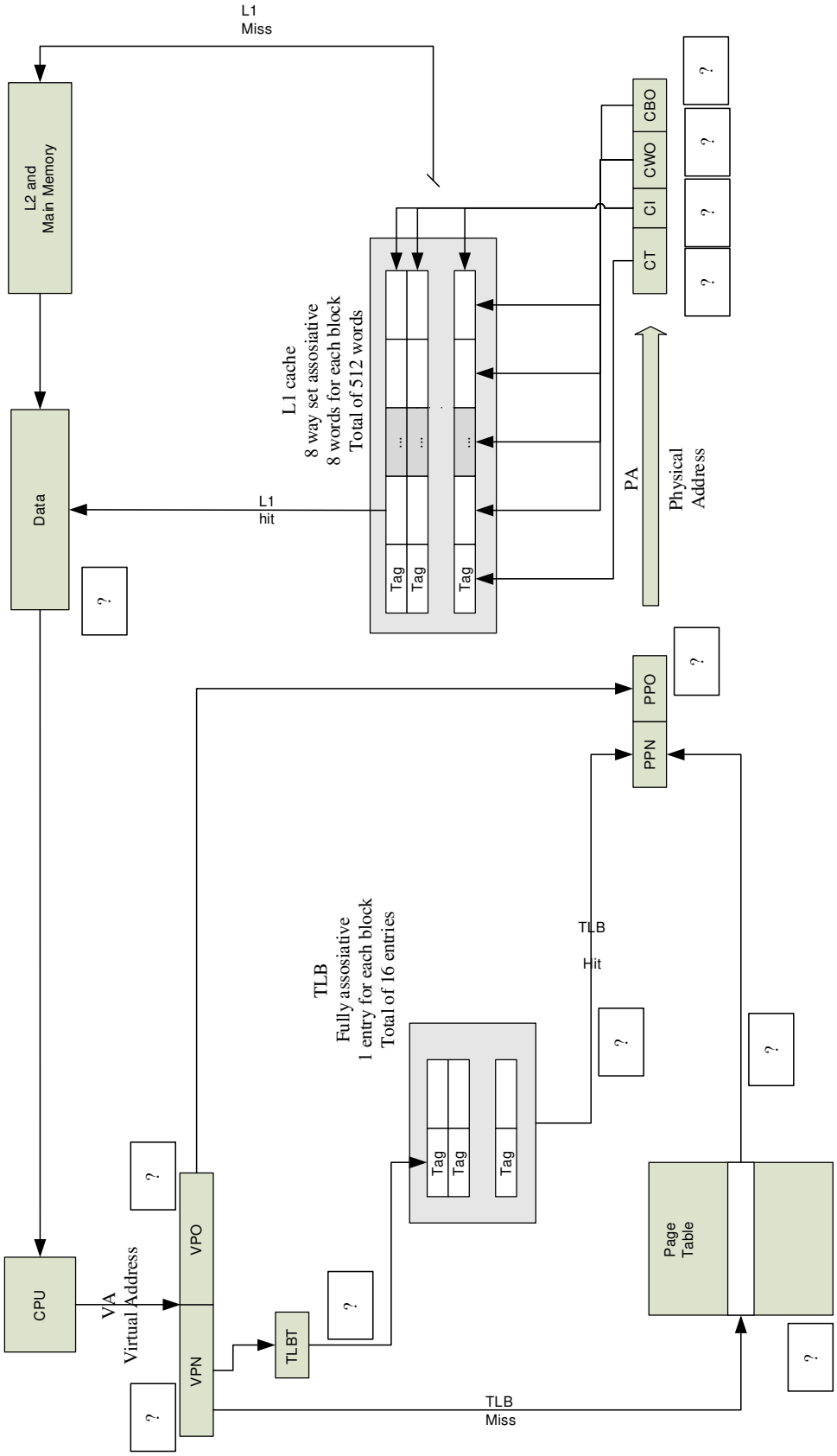
**İlan Tarihi: 31/05/2022 17:00**
**Teslim Tarihi: 06/06/2022 08:59**
**Telim Şekli: online.yildiz.edu.tr**

**Tek bir pdf dosyası olarak ödevinizi yükleyin.**
**Mail yolu ile ödev kabul edilmeyecektir.**

**Kopya ağır bir şekilde cezalandırılacaktır.**

**Soru 1)**

Şekil ile verilen virtual memory, TLB, page table, cache yapısı için Page size 1KB, Veri yolu genişliği 32 bit, Virtual address genişliği 20 bit, Physical address genişliği 18 bit olarak veriliyor. Buna göre sistem adres dönüşümünde oluşan adres parçalarının kaç bit uzunlukta olduğunu ve page table satır sayısını şekil üzerinde soru işaretli alanlara yazınız. (Hesaplamalarınızı gösterin)



**Soru 2)**

**Hazırlık:**

**CPU modelinizi https://www.computerhope.com/issues/ch000046.htm adresinde verilen şekilde belirleyiniz. Cevabınıza CPU modeline dair ekran çıktısını da ekleyiniz.**

**Linux için aşağıdaki komut da kullanılabilir:**

**$ lscpu**

**https://www.cpu-world.com/ adresinden CPU modelinizi arayarak, CPU'nun farklı seviyelerdeki veri ve komut önbellek (data and instruction cache) boyutlarını ve haritalama tiplerini not ediniz. Cevabınıza cache boyutlarına ve haritalama tiplerine dair ekran çıktısını da ekleyiniz.**

**Bundan sonraki işlemleri sanal makine üzerinde ubuntu kurulumu ile veya mevcut ise fiziki ubuntu kurulumu üzerinden devam ediniz.**

**gcc kurulumunu gerçekleyiniz.**

**$ sudo apt install build-essential**

**valgrind kurulumunu gerçekleyiniz.**

**$ sudo apt-get install valgrind**

Ekte verilen 1.c, 2.c ve 3.c dosyalarında matris çarpımına ilişkin farklı sıradaki döngü değişkenlerinin kullanımı ile oluşan hafıza ve komut adreslemeden doğan cache erişim durumlarını inceleyiniz.

Bunun için öncelikle kod derlemek için aşağıdaki adımları izleyiniz.

$ gcc 1.c -o 1.out

$ gcc 2.c -o 2.out

$ gcc 3.c -o 3.out

Oluşan çalıştırılabilir dosyaları valgrind'in cachegrind aracını kullanarak analiz ediniz.

$ valgrind --tool=cachegrind ./1.out

$ valgrind --tool=cachegrind ./2.out

$ valgrind --tool=cachegrind ./3.out

CPU'nun özelliklerini inceleyerek bulduğunuz cache boyut ve haritalama tiplerini dikkate alarak, boş bir cache ile başlandığı durumda, yürütülen kodlar için hem veri hem de komut açısından **L1 seviye** cahe hit ve miss oranlarını hesaplayınız. Elde ettiğiniz sonuçları valgrind çıktıları ile karşılaştırınız.

İşlemleri kod içindeki **dimension** sabiti 64 ve 256 değerlerindeyken tekrarlayınız.

Kodları çalıştırarak gerçekleşen matris çarpım işlem sürelerini not ediniz. Oluşan süreler ile cache hit miss oranlarını karşılaştırarak değerlendiriniz.