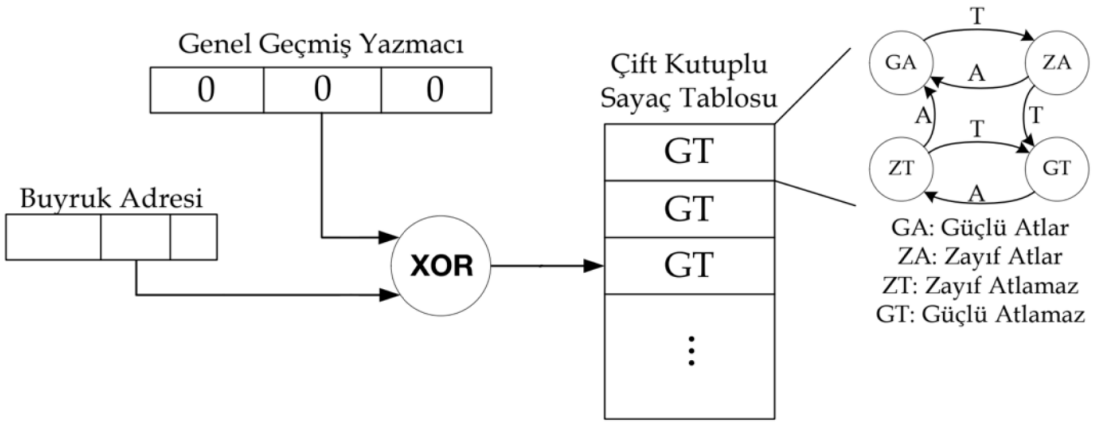
Bu ödevde hem gem5 simülatöründe hem de Verilog donanım tanımlama dilinde GShare dallanma öngörücüsü tasarlayacaksınız. *Combining Branch Predictors* (https://www.hpl.hp.com/techreports/Compaq-DEC/WRL-TN-36.pdf) adlı bildiriden yararlanabilirsiniz. Şekil 1'de örnek bir GShare dallanma öngörücü yapısı verilmiştir.



Şekil 1: Örnek GShare Dallanma Öngörücü Yapısı

[50 Puan] gem5 Simülatörü GShare Dallanma Öngörücüsü

gem5, C++ ve Python dillerinde yazılmış, birçok buyruk kümesi mimarisini (x86, ARM, MIPS, RISC-V vb.) simüle edebileceğiniz bir mimari simülatörü kütüphanesidir. (https://www.gem5.org) Siz de bu ödevde gem5'de hazır olarak bulunmayan gshare dallanma öngörücüsü birimi ekleyeceksiniz ve X86 mimarisi üzerinde coremark (https://github.com/eembc/coremark) benchmark programını çalıştırıp simüle ederek halihazırda var olan diğer dallanma öngörücüleri ile karşılaştıracaksınız.

Dallanma öngörücü birimini eklemeden önce gerekli bağlılıkları yüklemeniz ve kütüphaneyi doğru bir şekilde build etmeniz gerek.

Not: *Windows kullanıcısıysanız Ubuntu WSL (https://ubuntu.com/wsl) ya da VirtualBox (https://www.virtualbox.org), macOS kullanıcısıysanız yine VirtualBox ya da Docker (https://www.docker.com) gibi bir program üzerinde x86 ubuntu sanal makine kullanmanız ve ubuntu için olan adımları takip etmeniz gerekmektedir.*

Kütüphaneyi build etmek için gerekli araçları aşağıdaki gibi yükleyin:

```
sudo apt install build-essential
sudo apt install git
sudo apt install gcc g++
sudo apt install m4
sudo apt install scons
sudo apt install zlib1g
sudo apt install zlib1g-dev
sudo apt install python3-dev
```

gem5 kütüphanesini github'dan klonlayın ve `5fa484e2e02604ad3a5bf01f35ad1f97ca6d17b8` commit versiyonuna getirin (gem5 22.1.0.0 versiyonu) (https://github.com/gem5/gem5/archive/refs/tags/v22.1.0.0.zip adresinden de indirebilirsiniz):

```
git clone https://github.com/gem5/gem5
cd gem5
git checkout 5fa484e2e02604ad3a5bf01f35ad1f97ca6d17b8
python3 -m pip install -r requirements.txt
```

Kütüphaneyi elde ettikten sonra gem5 simülatörünü aşağıdaki gibi X86 buyruk kümesi mimarisi için build edin:

```
cd gem5
scons build/X86/gem5.opt -j1
```

Bu şekilde, gem5 simülatör programı (gem5.opt) *gem5/build/X86* dosya yolunda oluşmuş oldu. Burada daha hızlı build etmek için *-j1* ifadesinde "1" yerine daha fazla iş parçacığı (thread) atayabilirsiniz. Maksimum atayabileceğiniz iş parçacığı sayısını, terminalde *nproc* komutu ile öğrenebilirsiniz.

Bundan sonra kendi dallanma öngörücünüzü eklemek için ***gem5/src/cpu/pred*** dosya yoluna gitmeniz ve ödev kapsamında size verilen ***kasirgabp.hh*** ve ***kasirgabp.cc*** GShare dallanma öngörücü dosyalarını bu dosya yoluna eklemeniz gerekiyor. Daha sonra, ***gem5/src/cpu/pred/SConscript*** dosyasında *sim_objects* değişkenine *'KasirgaBP'* nesnesini ve *Source* satırını ekleyerek ***kasirgabp.cc***'yi build edilecekler listesine dahil edin:

```
sim_objects=[ ... 'KasirgaBP', ... ]

Source('kasirgabp.cc')
```

gem5/src/cpu/pred/BranchPredictor.py dosyasında ise aşağıdaki gibi dallanma öngörücüsü varsayılan parametrelerini verdiğiniz bir Python sınıfı eklemeniz gerekiyor:

```
class KasirgaBP(BranchPredictor):
    type = "KasirgaBP"
    cxx_class = "gem5::branch_prediction::KasirgaBP"
    cxx_header = "cpu/pred/kasirgabp.hh"

    globalPredictorSize = Param.Unsigned(8192, "Size of global predictor")
    globalCtrBits = Param.Unsigned(2, "Bits per counter")
```

Eklemeler bittikten sonra, *kasirgabp.cc* C++ kodunda GShare işlemi için ***TODO*** kısımlarında gerekli değişiklikleri yapın (*kasirgabp.cc* C++ kodunda yapılması gereken basit aritmetik işlemler sizin yapmanız için boş bırakılmıştır) ve *gem5* klasörüne giderek kütüphaneyi yeniden build edin:

```
cd gem5
scons build/X86/gem5.opt -j1
```

Kütüphanenin sorunsuz bir şekilde yeniden build olduğundan emin olduktan sonra, dallanma öngörücü performansını ölçmek için simülatörde bir program çalıştırmanız gerekiyor. *CoreMark* benchmarkını klonlayın (https://github.com/eembc/coremark/archive/refs/tags/v1.01.zip adresinden de indirebilirsiniz) ve programı X86 Linux için derleyin:

```
git clone https://github.com/eembc/coremark
cd coremark
make compile PORT_DIR=linux
```

CoreMark da sorunsuz bir şekilde derlendiyse artık aşağıdaki gibi gem5 simülatöründe *syscall emulation* modunda çalıştırabilirsiniz:

```
gem5/build/X86/gem5.opt \ ## X86 gem5.opt PATH'ini verin.
--outdir=gem5 \ ## çıktı dosyası PATH'i verin, stats.txt gibi çıktılar oluşacak
gem5/configs/example/se.py \ ## SE modu python dosyası PATH'ini verin
-c coremark/coremark.exe \ ## coremark.exe PATH'ini verin
--cpu-type=AtomicSimpleCPU \ ## basit bir CPU tipi
--maxinsts=10000000 \ ## maksimum 10 milyon buyruk çalıştırsın ve program dursun
--bp-type=KasirgaBP ## dallanma öngörücü ismi
```

Program çalıştıktan sonra verdiğiniz *outdir*'de ***stats.txt*** adlı dosya oluşacak. Bu dosya içinde "branch", "pred" gibi kelimeleri aratarak bu program için verilen dallanma öngörücüsüyle ne kadar doğru ya da yanlış tahmin yapıldığını görebilir ve eklediğiniz dallanma öngörücüsünün başarımını analiz edebilirsiniz. (***stats.txt***'deki değişkenlerin açıklamalarını yanlarında yorum olarak göreceksiniz.)

Not: *Çalıştırırken dosya yollarına dikkat edin, full path vermeye çalışın.*

gem5'de bulunan diğer dallanma öngörücüleriyle karşılaştırma yapmak için yukarıda gösterilen komutta **--bp-type=KasirgaBP** parametresini değiştirerek 2 farklı dallanma öngörücü için daha CoreMark'ı çalıştırın, stats.txt'leri elde edin ve bir yere kaydedin (gönderime de ekleyeceksiniz). (Bu 2 farklı dallanma öngörücüsünü, daha önce ekleme yaptığınız *gem5/src/cpu/pred/SConscript* dosyasında *sim_objects* değişkenlerinden seçebilirsiniz.) ***KasirgaBP*** dahil bu üç dallanma öngörücüsünün başarımlarını, ***stats.txt***'lerde gördüğünüz sayısal verilere dayalı olarak kısa cümlelerle özetleyerek karşılaştırın, *GShare* öngörü modelinin karşılaştırdığınız diğer dallanma öngörücülerine göre avantajlarını dezavantajlarını özetleyin ve <Soyad_Öğrenci Numarası>.pdf'e kaydedin.

[50 Puan] Verilog GShare Dallanma Öngörücüsü

Sizden, ekte verilen *kasirgabp.v* gshare dallanma öngörücüsünün içini -her saat vuruşunda GShare dallanma öngörüsü yapacak şekilde- doldurmanız bekleniyor. Tüm buyrukları dallanma buyruğu olarak varsayıp her saat vuruşunda dışarı öngörü çıkışını vermelisiniz.

Not: *Verilog tasarımı gem5 kısmından bağımsızdır. gem5 simülatörü için X86 buyruk kümesi mimarisi kullanılmaktadır fakat Verilog kısmının RISC-V buyruk kümesi için olduğunu ve hep 32 bit adresli 32 bit buyruklar olduğunu düşünebilirsiniz.*

Ödev Teslimi (Son Teslim Tarihi: 02.04.2023 23.59)

1-) kasirgabp.cc
2-) stats_kasirgabp.txt
3-) stats_<sectiginizdigerbp1>.txt
4-) stats_<sectiginizdigerbp2>.txt
5-) <Soyad_Öğrenci Numarası>.pdf
6-) kasirgabp.v
dosyalarını sıkıştırın ve <Soyad_Öğrenci Numarası>.zip olarak kaydederek https://uzak.etu.edu.tr'ye yükleyin.