

## Gem5 Installation

First I tried to get Gem5 works on Windows 10 using docker , it works but it was difficult to debug and it took more time because I'm not used to work with Docker that much.

The second platform I used the usual method, which using a fresh installed Linux distro for Gem5. I installed Linux Mint 22.1 on my laptop as main OS , and I followed these steps to get Gem5 works.

### 1- Make sure the system is updated using these commands :

```
sudo apt update  
sudo apt upgrade -y
```

### 2- Install Required Dependencies : Gem5 needs some software packages to build and work (Dependencies) . Install them using :

```
sudo apt install -y build-essential git m4 scons zlib1g zlib1g-dev libprotobuf-dev protobuf-compiler  
libprotoc-dev libgoogle-perftools-dev python3-dev python3-pip python-is-python3 git
```

### 3- Download Gem5 : get Gem5 source code using Git:

```
git clone https://github.com/gem5/gem5.git
```

Now we have gem5 directory and we can access using **cd gem5** .

There are two remained requirements to install "mypy and pre-commit" which are python library , we can install them using the command :

```
pip install -r requirements.txt
```

or by simply using apt :

```
sudo apt install python3-mypy pre-commit
```

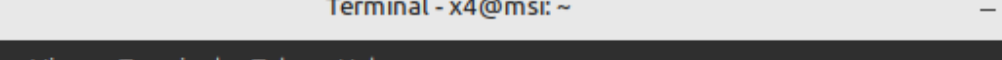
### - Building Gem5 using scons :

Gem5 directory includes all valid ISAs (Instruction Set Architectures) as followed :

**ARM**  
**NULL**  
**MIPS**  
**POWER**  
**RISCV**  
**SPARC**  
**X86**

Some of these ISAs have more than one type we can build , and we can use ALL to compiles Gem5 for all supported CPU architectures (X86, ARM, RISC-V, etc.). .

First : we use **nproc** command to display the number of available processing units. In my laptop I have 12 as shown.



```
Terminal - x4@msi: ~  
File Edit View Terminal Tabs Help  
x4@msi:~$ nproc  
12  
x4@msi:~$
```

I have to use more than one core in building process to make it faster .

In gem5 directory I executed this command  
scons build/[ISA]/gem5.opt -j [Number of cores]

I tried to build ALL using 9 cores by this command :

```
scons build/ALL/gem5.opt -j 9
```

It took about 2 hours to complete , and after using Gem5 I build another platform for Intel/AMD only by using

```
scons build/X86/gem5.opt -j 9
```

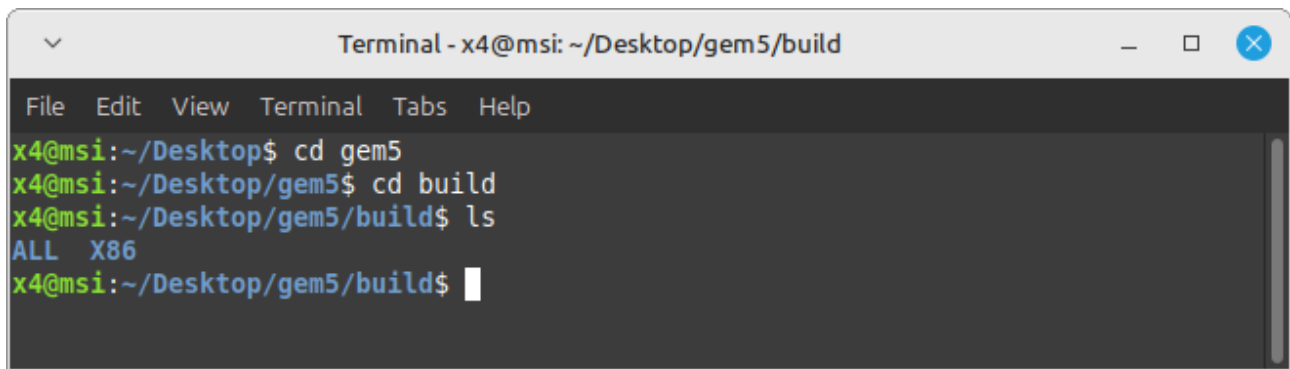
It took about 30 minutes to complete.

```

Terminal - x4@msi - /Desktop/xjem5
File Edit View Terminal Tabs Help
CXX X86/python/_m5/param_BaseSimpleCPU.cc -> .o
CXX X86/cpu/simple/AtomicSimpleCPU.py.cc -> .o
CXX X86/cpu/simple/NonCacheSimpleCPU.py.cc -> .o
CXX X86/cpu/simple/TimingSimpleCPU.py.cc -> .o
CXX X86/cpu/simple/probes/SimPoint.py.cc -> .o
50 Param m5.objects.SimPoint, SimPoint -> X86/python/_m5/param_SimPoint.cc
50 Param m5.objects.SimPoint, SimPoint -> X86/params/SimPoint.hh
CXX X86/python/_m5/param_SimPoint.cc -> .o
CXX src/cpu/simple/probes/simpoint.cc -> X86/cpu/simple/probes/simpoint.o
CXX X86/cpu/simple/probes/looppointanalysis.py.cc -> .o
50 Param m5.objects.LooppointAnalysis, LooppointAnalysis -> X86/python/_m5/param_LooppointAnalysis.cc
50 Param m5.objects.LooppointAnalysis, LooppointAnalysis -> X86/params/LooppointAnalysis.hh
TRACING -> X86/debug/LooppointAnalysis.hh
50 Param m5.objects.LooppointAnalysis, LooppointAnalysisManager -> X86/params/LooppointAnalysisManager.hh
50 Param m5.objects.LooppointAnalysis, LooppointAnalysisManager -> X86/python/_m5/param_LooppointAnalysisManager.cc
TRACING -> X86/debug/LooppointAnalysis.cc
CXX X86/debug/LooppointAnalysis.cc -> .o
CXX X86/python/_m5/param_LooppointAnalysis.cc -> .o
CXX X86/python/_m5/param_LooppointAnalysisManager.cc -> .o
CXX src/cpu/simple/probes/looppoint_analysis.cc -> X86/cpu/simple/probes/looppoint_analysis.o
CXX X86/debug/branchpredictor.py.cc -> .o
50 Param m5.objects.BranchPredictor, BranchPredictor -> X86/python/_m5/param_BranchPredictor.cc
CXX X86/python/_m5/param_BranchPredictor.cc -> .o
50 Param m5.objects.BranchPredictor, IndirectPredictor -> X86/python/_m5/param_IndirectPredictor.cc
CXX X86/python/_m5/param_IndirectPredictor.cc -> .o
50 Param m5.objects.BranchPredictor, SimpleIndirectPredictor -> X86/python/_m5/param_SimpleIndirectPredictor.cc
50 Param m5.objects.BranchPredictor, BranchTargetBuffer -> X86/python/_m5/param_BranchTargetBuffer.cc
50 Param m5.objects.BranchPredictor, SimpleIndirectPredictor -> X86/params/SimpleIndirectPredictor.hh
CXX X86/python/_m5/param_BranchTargetBuffer.cc -> .o
CXX X86/python/_m5/param_SimpleIndirectPredictor.cc -> .o
50 Param m5.objects.BranchPredictor, SimpleBTB -> X86/python/_m5/param_SimpleBTB.cc
50 Param m5.objects.BranchPredictor, SimpleBTB -> X86/params/SimpleBTB.hh
50 Param m5.objects.BranchPredictor, BTBIndexingPolicy -> X86/params/BTBIndexingPolicy.hh
50 Param m5.objects.BranchPredictor, BTBSetAssociative -> X86/params/BTBSetAssociative.hh
CXX X86/python/_m5/param_BTBTIndexingPolicy.cc -> .o
50 Param m5.objects.BranchPredictor, BTBTIndexingPolicy -> X86/python/_m5/param_BTBTIndexingPolicy.cc
CXX X86/python/_m5/param_BTBTSetAssociative.cc -> .o
CXX X86/python/_m5/param_BTBTIndexingPolicy.cc -> .o
50 Param m5.objects.BranchPredictor, ReturnAddrStack -> X86/python/_m5/param_ReturnAddrStack.cc
CXX X86/python/_m5/param_ReturnAddrStack.cc -> .o
50 Param m5.objects.BranchPredictor, LocalBP -> X86/python/_m5/param_LocalBP.cc
50 Param m5.objects.BranchPredictor, LocalBP -> X86/params/LocalBP.hh
50 Param m5.objects.BranchPredictor, TournamentBP -> X86/python/_m5/param_TournamentBP.cc
CXX X86/python/_m5/param_LocalBP.cc -> .o
50 Param m5.objects.BranchPredictor, TournamentBP -> X86/params/TournamentBP.hh
CXX X86/python/_m5/param_TournamentBP.cc -> .o
50 Param m5.objects.BranchPredictor, B1ModeBP -> X86/python/_m5/param_B1ModeBP.cc
50 Param m5.objects.BranchPredictor, B1ModeBP -> X86/params/B1ModeBP.hh
50 Param m5.objects.BranchPredictor, TAGBase -> X86/python/_m5/param_TAGBase.cc
CXX X86/python/_m5/param_B1ModeBP.cc -> .o
50 Param m5.objects.BranchPredictor, TAG -> X86/python/_m5/param_TAG.cc
50 Param m5.objects.BranchPredictor, TAGBase -> X86/params/TAGBase.hh
CXX X86/python/_m5/param_TAGBase.cc -> .o
50 Param m5.objects.BranchPredictor, TAG -> X86/params/TAG.hh
CXX X86/python/_m5/param_TAG.cc -> .o

```

We can see the compiled ISAs inside build sub-directory in Gem5.

A terminal window titled "Terminal - x4@msi: ~/Desktop/gem5/build". The window has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The terminal content shows the following commands and output:

```
x4@msi:~/Desktop$ cd gem5
x4@msi:~/Desktop/gem5$ cd build
x4@msi:~/Desktop/gem5/build$ ls
ALL  X86
x4@msi:~/Desktop/gem5/build$
```

By these steps we have Gem5 ready to use on our system.