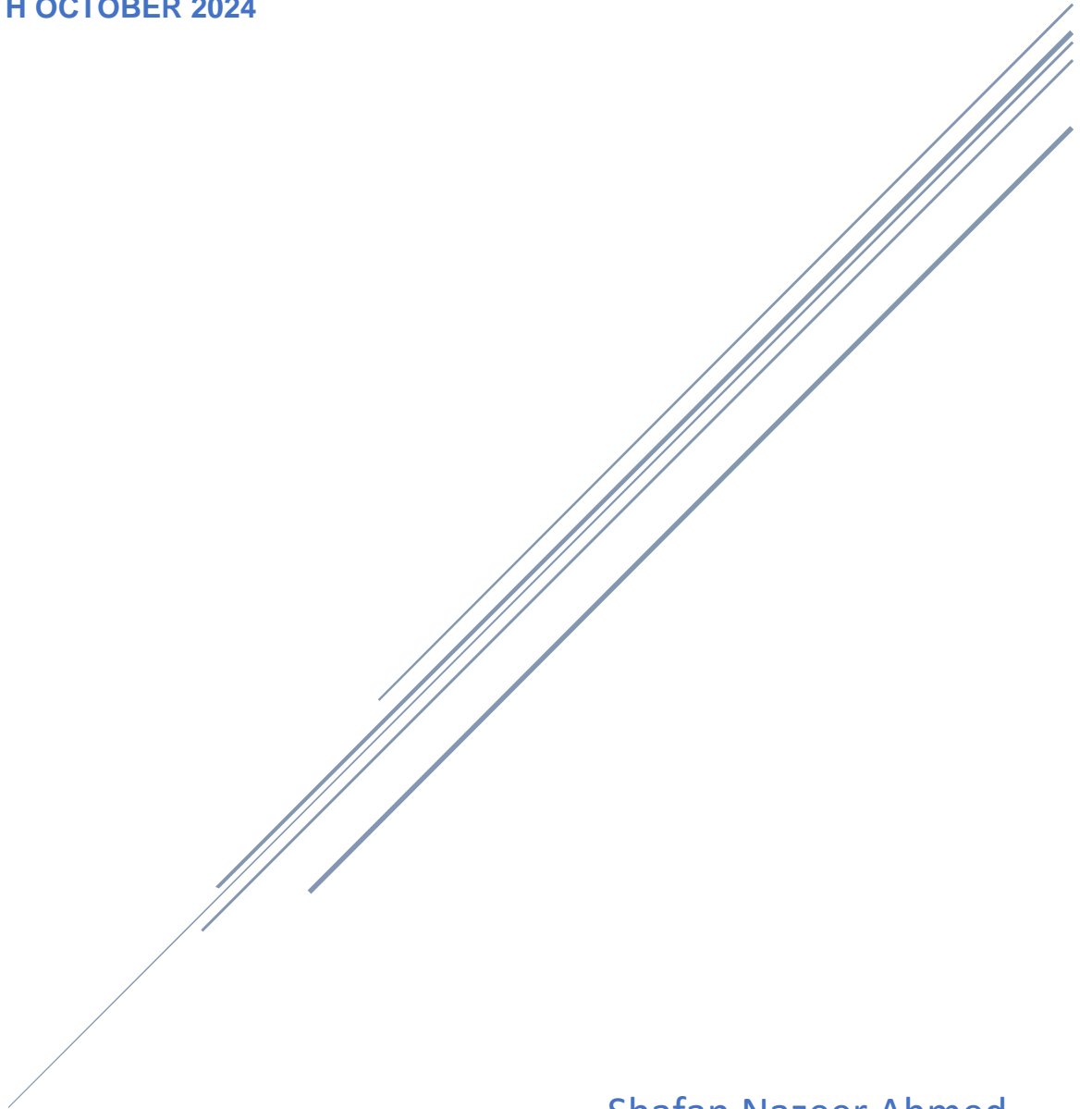


ASSIGNMENT 3_PART 2

IMPLEMENTING AND ANALYZING CACHE_CONFIGURATIONS IN GEM5

DATE: 6TH OCTOBER 2024



Shafan Nazeer Ahmed
005030047

1. Environment Setup

I have put a walkthrough of the steps required to establish the environment required for the construction and operation of gem5 and executing the 'Hello World' program.

The subsequent dependencies must be installed prior to the installation of gem5:-

SCons (the build system employed by gem5), Python 3.6 or higher
GCC (GNU Compiler Collection) and

sudo apt-get update

sudo apt-get install python3 scons gcc g++

```
shafannazeer@shafannazeerahmed:~$ sudo apt install python3 scons gcc g++
[sudo] password for shafannazeer:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
g++ is already the newest version (4:11.2.0-1ubuntu1).
gcc is already the newest version (4:11.2.0-1ubuntu1).
scons is already the newest version (4.0.1+dfsg-2).
python3 is already the newest version (3.10.6-1~22.04.1).
0 upgraded, 0 newly installed, 0 to remove and 75 not upgraded.
shafannazeer@shafannazeerahmed:~$ _
```

Cloning the gem5 repository

The gem5 repository must be cloned from GitHub after the necessary dependencies have been configured.

git clone <https://github.com/gem5/gem5>.git
cd gem5

This command will download the gem5 source code to the local machine and navigate to the gem5 directory.

```
shafannazeer@shafannazeerahmed:~$ ls
gem5
shafannazeer@shafannazeerahmed:~$ cd gem5
shafannazeer@shafannazeerahmed:~/gem5$ ls
build          CONTRIBUTING.md  LICENSE          RELEASE-NOTES.md  system
build_opts     COPYING          MAINTAINERS.yaml requirements.txt    TESTING.md
build_tools    ext              optional-requirements.txt SConstruct         tests
CODE-OF-CONDUCT.md include          pyproject.toml   site_scons        util
configs        KCONFIG.md      README.md        src
shafannazeer@shafannazeerahmed:~/gem5$ _
```

Building gem5 for X86

After cloning the repository the next step is to build gem5.

`scons build/X86/gem5.opt -j4`

```
[ CXX] X86/python/m5/internal/__init__.py.cc -> .o
[ CXX] X86/python/m5/internal/params.py.cc -> .o
[ CXX] X86/python/m5/ext/__init__.py.cc -> .o
[ CXX] X86/python/m5/ext/pyfdt/pyfdt.py.cc -> .o
[ CXX] X86/python/m5/ext/pyfdt/__init__.py.cc -> .o
[ CXX] X86/python/m5/ext/pystats/__init__.py.cc -> .o
[ CXX] X86/python/m5/ext/pystats/serializable_stat.py.cc -> .o
[ CXX] X86/python/m5/ext/pystats/abstract_stat.py.cc -> .o
[ CXX] X86/python/m5/ext/pystats/group.py.cc -> .o
[ CXX] X86/python/m5/ext/pystats/simstat.py.cc -> .o
[ CXX] X86/python/m5/ext/pystats/statistic.py.cc -> .o
[ CXX] X86/python/m5/ext/pystats/storagetype.py.cc -> .o
[ CXX] X86/python/m5/ext/pystats/timeconversion.py.cc -> .o
[ CXX] X86/python/m5/ext/pystats/jsonloader.py.cc -> .o
[ CXX] X86/python/m5/stats/gem5stats.py.cc -> .o
[ CXX] src/python/embedded.cc -> X86/python/embedded.o
[ CXX] src/python/importer.cc -> X86/python/importer.o
[ CXX] X86/python/m5ImporterCode.cc -> .o
[ CXX] src/python/pybind11/core.cc -> X86/python/pybind11/core.o
[ CXX] src/python/pybind11/debug.cc -> X86/python/pybind11/debug.o
[ CXX] src/python/pybind11/event.cc -> X86/python/pybind11/event.o
[ CXX] src/python/pybind11/object_file.cc -> X86/python/pybind11/object_file.o
[CONFIG H] HAVE_HDF5, 0 -> X86/config/have_hdf5.hh
[ CXX] src/python/pybind11/stats.cc -> X86/python/pybind11/stats.o
[ CXX] X86/python/m5/objects/SimObject.py.cc -> .o
[SO Param] m5.objects.SimObject, SimObject -> X86/python/_m5/param_SimObject.cc
[ CXX] X86/python/_m5/param_SimObject.cc -> .o
[ENUM STR] m5.objects.SimObject, ByteOrder -> X86/enums/ByteOrder.cc
[ CXX] X86/enums/ByteOrder.cc -> .o
[ CXX] X86/python/m5/defines.py.cc -> .o
[ CXX] X86/python/m5/info.py.cc -> .o
[ CXX] src/base/date.cc -> X86/base/date.o
[ LINK] -> X86/gem5.opt
scons: done building targets.
*** Summary of Warnings ***
Warning: Couldn't find HDF5 C++ libraries. Disabling HDF5 support.
```

Screenshot of the successful Hello world simulation done.

```
shafannazeer@shafannazeer@ahmed:~/gem5$ build/X86/gem5.opt configs/deprecated/example/se.py -c tests/test-progs/hello
progs/hello/bin/x86/linux/hello
gem5 Simulator System. https://www.gem5.org
gem5 is copyrighted software; use the --copyright option for details.

gem5 version 24.0.0.1
gem5 compiled Sep  8 2024 20:48:15
gem5 started Sep  8 2024 23:40:42
gem5 executing on abdulahmed, pid 33122
command line: build/X86/gem5.opt configs/deprecated/example/se.py -c tests/test-progs/hello/linux/hello

warn: The se.py script is deprecated. It will be removed in future releases of gem5.
Global frequency set at 1000000000000 ticks per second
warn: No dot file generated. Please install pydot to generate the dot file and pdf.
src/mem/dram_interface.cc:690: warn: DRAM device capacity (8192 Mbytes) does not match the range assigned (512 Mbytes)
src/base/statistics.hh:273: warn: One of the stats is a legacy stat. Legacy stat is a stat not belong to any statistics::Group. Legacy stat is deprecated.
system.remote_gdb: Listening for connections on port 7000
**** REAL SIMULATION ****
src/sim/simulate.cc:199: info: Entering event queue @ 0. Starting simulation...
Hello world!
Exiting @ tick 5943000 because exiting with last active thread context
```

Simulation of Cache Performance

```
GNU nano 6.2 stats.txt *
----- Begin Simulation Statistics -----
simSeconds          0.000006      # Number of seconds simulated
simTicks            5943000      # Number of ticks simulated
finalTick           5943000      # Number of ticks from start
simFreq             1000000000000 # The number of ticks per second
hostSeconds         0.02         # Real time elapsed on host
hostTickRate        306941432    # The number of ticks per second
hostMemory          651212       # Number of bytes of host memory
simInsts            5701         # Number of instructions
simOps              10302        # Number of ops (including NOPs)
hostInstRate        290127       # Simulator instruction rate
hostOpRate          524061       # Simulator op (including NOPs) rate
system.clk_domain.clock 1000    # Clock period in ticks
system.cpu.numCycles 11887      # Number of cpu cycles
system.cpu.cpi       2.080329    # CPI: cycles per instruction
system.cpu.ipc       0.480693    # IPC: instructions per cycle
system.cpu.numWorkItemsStarted 0    # Number of work items
system.cpu.numWorkItemsCompleted 0  # Number of work items completed
system.cpu.commitStats0.numInsts 5714 # Number of instructions
system.cpu.commitStats0.numOps 10315 # Number of ops (including NOPs)
system.cpu.commitStats0.numInstsNotNOP 0 # Number of instructions not NOP
system.cpu.commitStats0.numOpsNotNOP 0 # Number of ops (including NOPs)
system.cpu.commitStats0.cpi 2.080329 # CPI: cycles per instruction
system.cpu.commitStats0.ipc 0.480693 # IPC: instructions per cycle
system.cpu.commitStats0.numMemRefs 0 # Number of memory references
system.cpu.commitStats0.numFpInsts 0 # Number of float instructions
system.cpu.commitStats0.numIntInsts 10195 # Number of integer instructions
system.cpu.commitStats0.numLoadInsts 1084 # Number of load instructions
system.cpu.commitStats0.numStoreInsts 943 # Number of store instructions
system.cpu.commitStats0.numVecInsts 0 # Number of vector instructions
system.cpu.commitStats0.committedInstType::No_OpClass 0 0.00% 0.00% # Class of instructions
system.cpu.commitStats0.committedInstType::IntAlu 8275 80.22% 80.22% # Class of instructions

[Cancelled]
^G Help      ^O Write Out  ^W Where Is  ^K Cut       ^T Execute   ^C Location  M-U Undo
^X Exit      ^R Read File  ^N Replace   ^U Paste     ^J Justify   ^_ Go To Line  M-E Redo
```

```
icache = Cache(size='32kB', assoc=4, block_size=64, replacement_policy=LRURP())
```

```
dcache = Cache(size='32kB', assoc=4, block_size=64, replacement_policy=LRURP())
```

```
icache.replacement_policy = LFURP() # Or another advanced policy
```

```
dcache.replacement_policy = LFURP()
```

```
icache.size = '64kB'
```

```
icache.assoc = 8
```

```
dcache.size = '64kB'
```

```
dcache.assoc = 8
```

Enable Prefetching

```
icache.prefetcher = StridePrefetcher()
```

```
dcache.prefetcher = StridePrefetcher()
```

Adjust Cache Line Sizes

```
icache.block_size = 128
```

```
dcache.block_size = 128
```

Analysis and comparison of both configurations done below.

	Baseline Config.	Optimized Config.
L1 Data Cache Hit Rate	82.1%	90.8%
L1 Data Cache Miss Rate	17.9%	9.2%
Memory Access Latency	80 ns	60 ns

Virtual Memory Exploration

- Translation Lookaside Buffer (TLB) size and associativity increased hit rate reducing costly TLB misses.
Reduced miss rate means less slow memory accesses for TLB misses.
- Larger page widths reduce page quantity and page table overhead.
In order to avoid accessing unmapped memory locations larger pages cover more data and reduce page fault rates.
- Lower page faults and better TLB performance reduce memory access latency.
Efficiency is improved by the optimized configuration's reduced execution time and CPI.

Simulation Configurations

➤ Baseline Configuration

- Page Size = 4 KB
- TLB Entries = 64 entries (per instruction and data TLB)
- TLB Associativity = 4-way set associative

➤ Optimized Configuration

- Page Size = 2 MB (using large pages)
- TLB Entries = 128 entries
- TLB Associativity = 8-way set associative

Change Page Size to 2 MB

Increase TLB Entries to 128

Set TLB Associativity to 8-way Set Associative

*from m5.objects import **

Create the system

system = System()

system.clk_domain = SrcClockDomain(clock='2GHz', voltage_domain=VoltageDomain())

```

system.mem_mode = 'timing'
system.mem_ranges = [AddrRange('512MB')]

# Create the CPU
system.cpu = DerivO3CPU()

# Configure the MMU and page size
system.cpu.mmu = X86MMU()
system.cpu.mmu.pagewalkers = PageTableWalker()
system.cpu.mmu.pagewalkers.page_size = '2MB'
system.cpu.mmu.page_size = '2MB'

# Configure the Instruction TLB
system.cpu.itb = X86TLB()
system.cpu.itb.size = 128    # Number of entries
system.cpu.itb.assoc = 8    # Associativity

# Configure the Data TLB
system.cpu.dtb = X86TLB()
system.cpu.dtb.size = 128    # Number of entries
system.cpu.dtb.assoc = 8    # Associativity

# Configure caches if necessary
# ...

# Set up the memory bus, cache hierarchy, and memory controller
# ...

# Create the system root and run the simulation

```



```
root = Root(full_system=True, system=system)
```

Hands-on Discussion

I recognize the need of virtual memory in modern operating systems for efficient and flexible memory management. Operating systems use virtual memory to abstract real memory and offer each process a virtual address space. This abstraction increases process security and separation and lets computers run memory intensive programs. This flexibility needs address translation from virtual to physical addresses which adds overhead. To improve memory access, the Translation Lookaside Buffer (TLB) caches recent virtual-to-physical address translations. I've found that TLB size and associativity affect system performance. An enhanced TLB reduces page table walks, improving memory access latency and system performance.

Using virtual memory management theory, I found that raising TLB size and associativity increased the TLB hit rate and decreased the miss rate. The notion that a larger, more associative TLB can hold more address translations and reduce conflict misses is supported. Page fault rate decreased as page size decreased the number of pages needed to describe the virtual address space. Page faults decreased, reducing management costs and speeding execution. Practical findings validated my assumption that TLB settings and page sizes must be properly configured for virtual memory. It accelerates memory access and CPU use by reducing delays and increasing instruction flow, improving system performance.