Gem5 Execution

After I have Gem5 with ALL and x86 ISAs ready , I tried some basic testing for both using simple C codes .

I wrote 4 basics codes and compiled them in sub-directory "Gem5Examples"

1. hello.c

```c
#include <stdio.h>
int main() {
    printf("Hello Abdullah !\nThis is your greeting code from Gem5");
    return 0;
}
```

2.loop.c (Simple loop)

```c
#include <stdio.h>
int main() {
    for (int i = 0; i < 5; i++) {
        printf("Loop iteration: %d\n", i);
    }
    return 0;
}
```

3. sum.c (Arthimatic Sum Operation)

```c
#include <stdio.h>
int main() {
    int a = 5, b = 3;
    printf("Sum: %d\n", a + b);
    return 0;
}
```

4.array.c (Array)

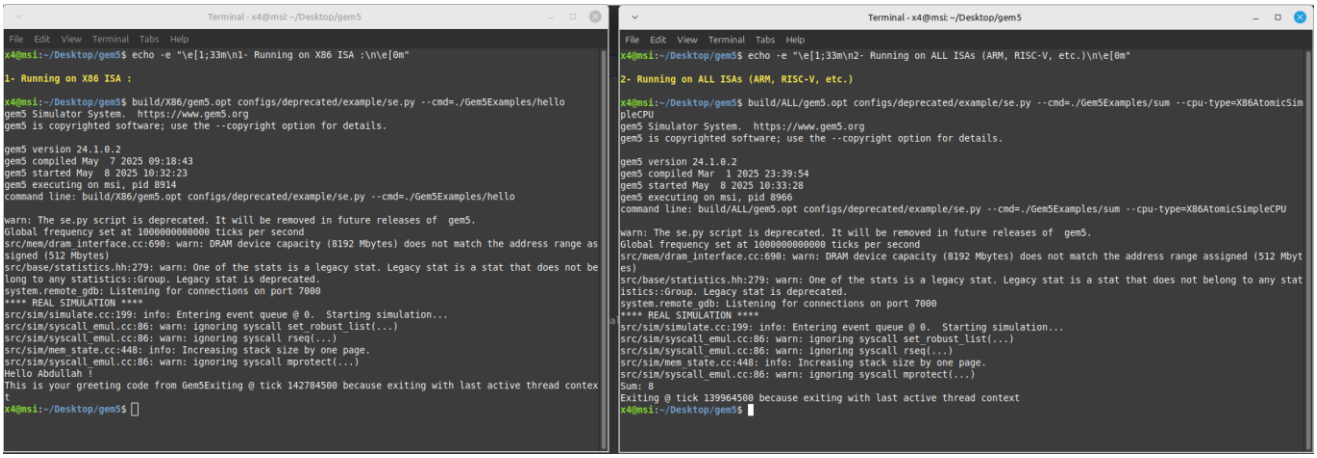
```c
#include <stdio.h>
int main() {
    int arr[3] = {10, 20, 30};
    printf("Array element: %d\n", arr[1]);
    return 0;
}
```

Then I compiled them with same names as followed :

```
gcc -static hello.c -o hello
gcc -static loop.c -o loop
gcc -static sum.c -o sum
gcc -static array.c -o array
```

After that I tested each execution for these codes in both ALL(With different cpu options) and x86.

I use System Emulation script "configs/deprecated/example/se.py" which use to simulate a single process or small programm.



1- Running on X86 ISA :

build/X86/gem5.opt configs/deprecated/example/se.py --cmd=./Gem5Examples/hello

build/X86/gem5.opt configs/deprecated/example/se.py --cmd=./Gem5Examples/loop

build/X86/gem5.opt configs/deprecated/example/se.py --cmd=./Gem5Examples/sum

build/X86/gem5.opt configs/deprecated/example/se.py --cmd=./Gem5Examples/array

2- Running on ALL ISAs (ARM, RISC-V, etc.)

build/ALL/gem5.opt configs/deprecated/example/se.py --cmd=./Gem5Examples/hello --cpu-type=ArmAtomicSimpleCPU

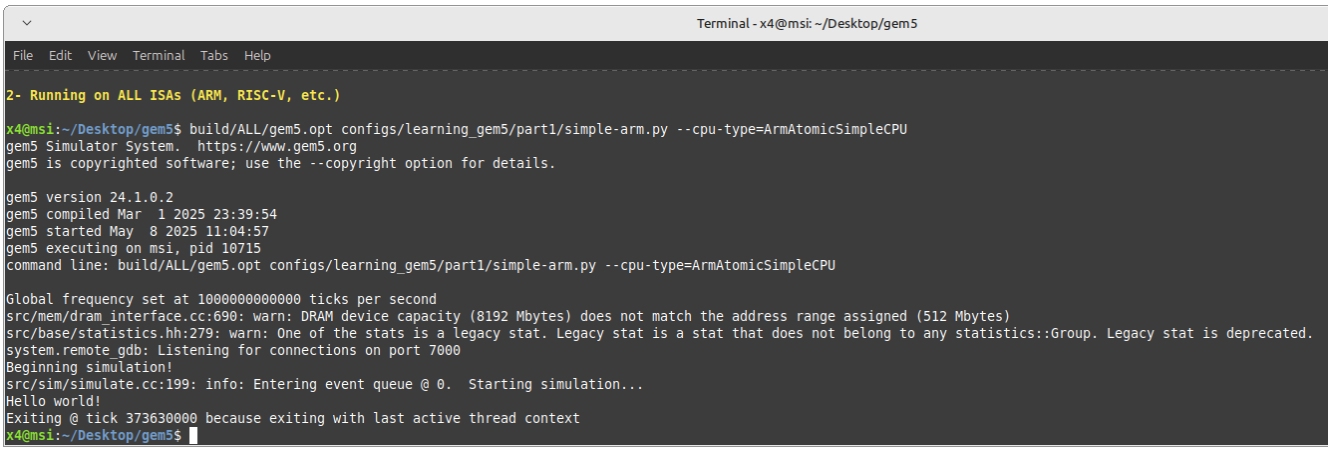
build/ALL/gem5.opt configs/deprecated/example/se.py --cmd=./Gem5Examples/loop --cpu-type=RiscvAtomicSimpleCPU

build/ALL/gem5.opt configs/deprecated/example/se.py --cmd=./Gem5Examples/sum --cpu-type=X86AtomicSimpleCPU

build/ALL/gem5.opt configs/deprecated/example/se.py --cmd=./Gem5Examples/array --cpu-type=ArmAtomicSimpleCPU

Some codes didn't work using ARM or Risc we had errors like "Segmentation fault (core dumped)" , because the codes compiled to work with x86 , but i did it as an expermintal test .

I used the example of ARM:



build/ALL/gem5.opt configs/learning_gem5/part1/simple-arm.py --cpu-type=ArmAtomicSimpleCPU

examples included in configs/learning_gem5/

part1:

caches.py simple-arm.py simple.py simple-riscv.py two_level.py

part2:

hello_goodbye.py run_simple.py simple_cache.py simple_memobj.py

part3:

msi_caches.py ruby_caches_MI_example.py ruby_test.py simple_ruby.py test_caches.py

we can use them to practice small executions and simulations using Gem5

deprecated/example , the main examples we can tested :

se.py (System Emulation) is the most commonly used configuration script in Gem5 because:

Simple & Fast: It simulates a single process (like running a single C program) without booting a full OS.

Good for Testing: Perfect for running small programs (like your 5 C examples).

Flexible: Allows tweaking CPU, cache, and memory settings easily.

fs.py (Full System Simulation)

Boots a full Linux OS (like Ubuntu) inside Gem5.

When to use:

Need to test multi-process programs (e.g., fork(), threads).

Running real applications (e.g., Apache, Redis).

Testing device drivers or kernel modules.