Assignment 1

*Haiyu Wang*

1. Configuration File:

x86\_vs\_arm.py

# Assignment 1, Haiyu Wang

# Usage: like $GEM5/build/X86/gem5.opt --outdir="daxpy\_x86"

# x86\_vs\_arm.py --prog="daxpy"

#

# Set up simulator with GEM5, run programs and test them

import m5

from m5.objects import \*

import os

gem5\_path = os.environ["GEM5"]

#option parser and specify the program

import optparse

parser = optparse.OptionParser()

parser.add\_option("--prog", type="str", default=None);

(options, args) = parser.parse\_args()

program = options.prog

system = System() #instantiate system

system.clk\_domain = SrcClockDomain() #set clock domain

system.clk\_domain.voltage\_domain = VoltageDomain() #set voltage domain

# set clock condition

isa = m5.defines.buildEnv['TARGET\_ISA']

if isa == "x86":

system.clk\_domain.clock = '1GHz'

elif isa == "arm":

system.clk\_domain.clock = '1.2GHz'

# use timing mode and set memory range

system.mem\_mode = 'timing'

system.mem\_ranges = [AddrRange('512MB')]

#set cpu and membus

system.cpu = TimingSimpleCPU()

system.membus = SystemXBar()

# set cache

system.cpu.icache\_port = system.membus.slave

system.cpu.dcache\_port = system.membus.slave

# connect PIO and interrupts ports to membus

system.cpu.createInterruptController()

if isa == 'x86':

system.cpu.interrupts[0].pio = system.membus.master

system.cpu.interrupts[0].int\_master = system.membus.slave

system.cpu.interrupts[0].int\_slave = system.membus.master

system.system\_port = system.membus.slave

# set memory controller

system.mem\_ctrl = DDR3\_1600\_8x8()

system.mem\_ctrl.range = system.mem\_ranges[0]

system.mem\_ctrl.port = system.membus.master

# set up process

process = Process()

apps\_path = "/project/linuxlab/gem5/test\_progs"

if program == "daxpy" and isa == "x86":

process.cmd = [apps\_path + '/daxpy/daxpy\_x86']

elif program == "daxpy" and isa == "arm":

process.cmd = [apps\_path + '/daxpy/daxpy\_arm']

elif program == "queens" and isa == "x86":

process.cmd = [apps\_path + '/queens/queens\_x86']

process.cmd += ["10 -c"]

elif program == "queens" and isa == "arm":

process.cmd = [apps\_path + '/queens/queens\_arm']

process.cmd += ["10 -c"]

system.cpu.workload = process

system.cpu.createThreads()

# create root object

root = Root(full\_system = False, system = system)

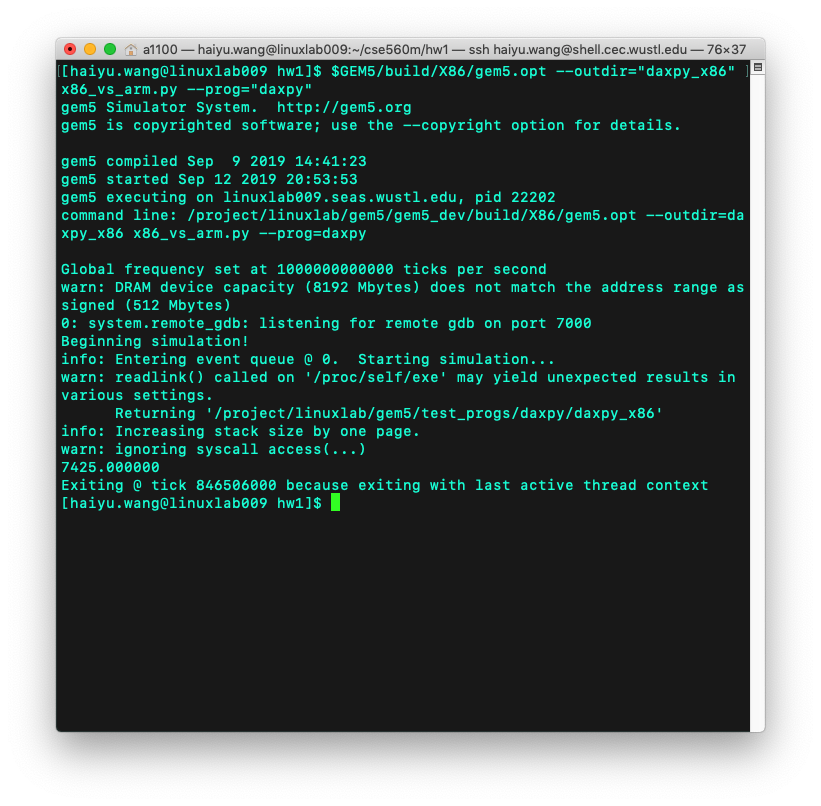
m5.instantiate()

print ("Beginning simulation!")

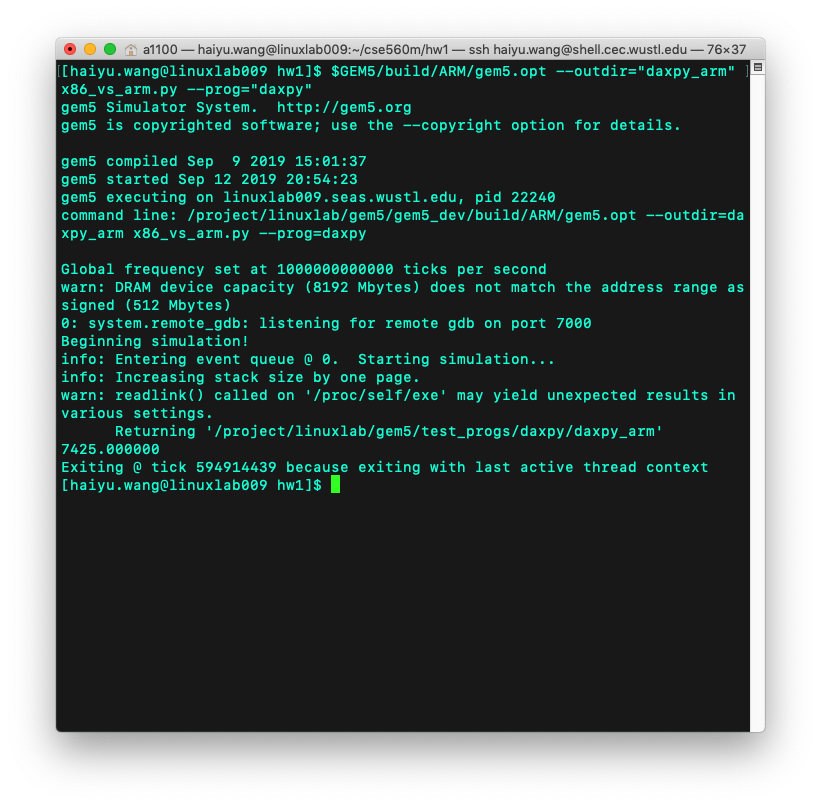
exit\_event = m5.simulate()

print('Exiting @ tick %i because %s' % (m5.curTick(), exit\_event.getCause()))

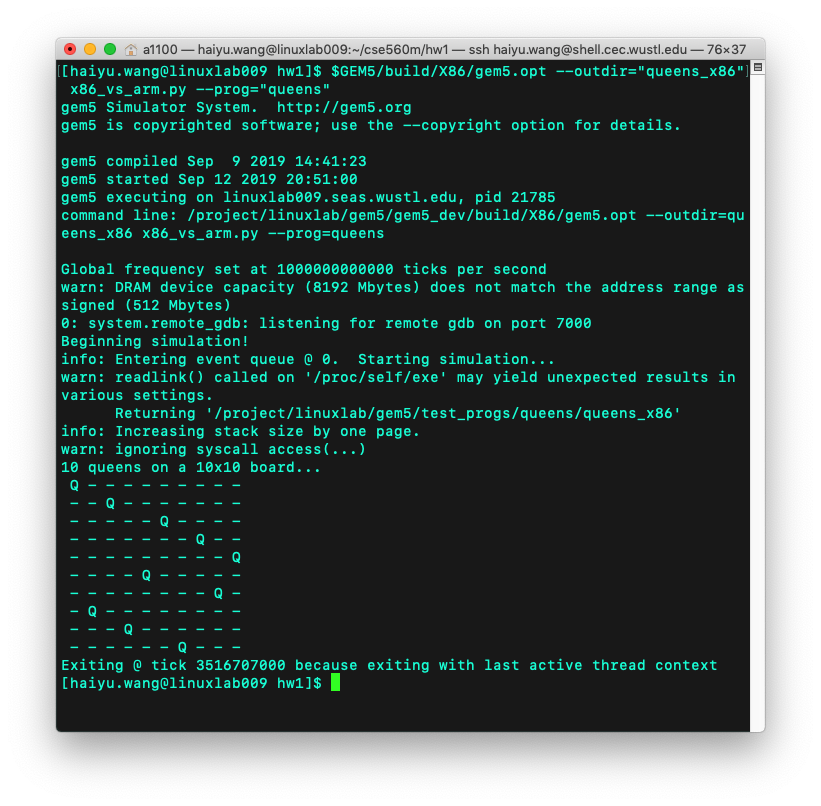
1. Outputs
   1. Program: daxpy\_x86



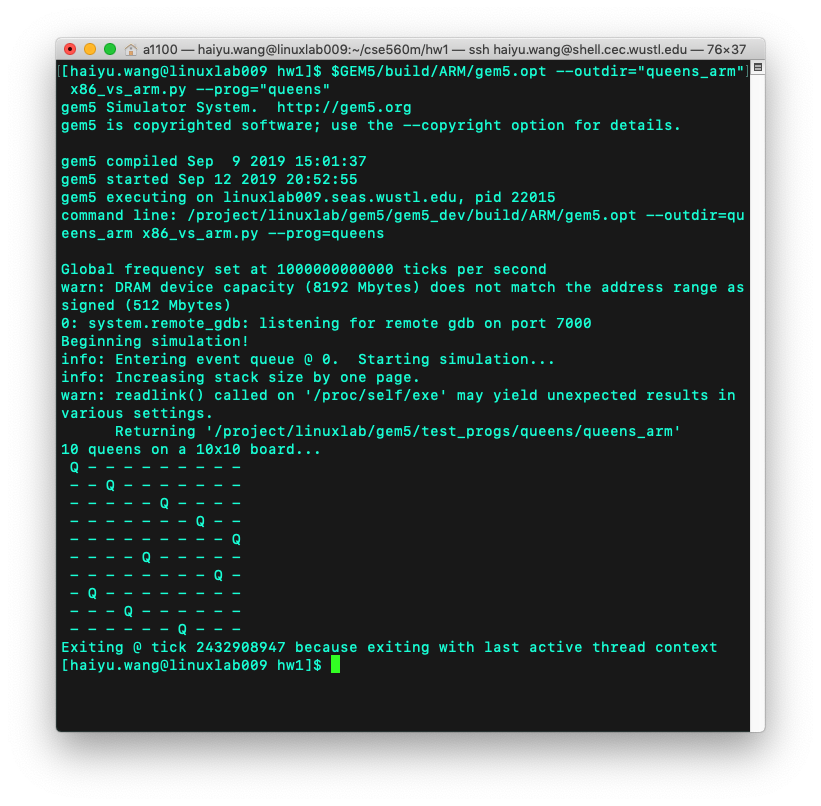
* 1. Program: daxpy\_arm



* 1. Program: queens\_x86



* 1. Program: queens\_arm



1. Response
   1. Confirmation of the execution time
      1. Program daxpy\_x86

|  |  |
| --- | --- |
| sim\_seconds | 0.000847 |
| Number of instructions | 10233 (sim\_insts) |
| Clock cycles | 846506 (system.cpu.numCycles) |
| CPI | 846505/10233 = 82.72 |
| tclk | 1000/1000000000000 = 0.000000001  (system.clk\_domain.clock/sim\_freq) |

texecution\_time = (number of instructions) × CPI × tclk

= 10233×82.72×0.000000001 ≈ 0.000847

* + 1. Program daxpy\_arm

|  |  |
| --- | --- |
| sim\_seconds | 0.000595 |
| Number of instructions | 9361 |
| Clock cycles | 714183 |
| CPI | 714183/9361 = 76.29 |
| tclk | 833/1000000000000 = 0.000000000833 |

texecution\_time = (number of instructions) × CPI × tclk

= 9361×76.29×0.000000000833 ≈ 0.000595

* + 1. Program queens\_x86

|  |  |
| --- | --- |
| sim\_seconds | 0.003517 |
| Number of instructions | 41261 |
| Clock cycles | 3516707 |
| CPI | 3516707/41261 = 85.23 |
| tclk | 1000/1000000000000 = 0.000000001 |

texecution\_time = (number of instructions) × CPI × tclk

= 41261×85.23×0.000000001 ≈ 0.003517

* + 1. Program queens\_arm

|  |  |
| --- | --- |
| sim\_seconds | 0.002433 |
| Number of instructions | 37778 |
| Clock cycles | 2920659 |
| CPI | 2920659/37778 = 77.31 |
| tclk | 833/1000000000000 = 0.000000000833 |

texecution\_time = (number of instructions) × CPI × tclk

= 37778×77.31×0.000000000833 ≈ 0.002433

With the calculation above, all of the sim\_seconds are the same as the texecution\_time, so we can confirm the execution time are sim\_seconds.

* 1. Plots

Note: mean is the average time of executing 10000 instructions.

* + 1. Plot: x86 execution time



* + 1. Plot: arm execution time



* + 1. Plot: speedup

Since the execution time of ARM ISA is shorter than that of x86 ISA, ARM ISA is faster.

|  |  |
| --- | --- |
|  | Speedup |
| daxpy | (1/0.000595 – 1/0.000847) ÷ (1/0.000847) × 100% = 42.35% |
| queens | (1/0.002433 – 1/0.003517) ÷ (1/0.003517) × 100% = 44.55% |
| mean | (1/0.000640 – 1/0.000840) ÷ (1/0.000840) × 100% = 31.25% |

