Nirbhay sharma(B19CSE114) - Lab-1

Part1

Parameters	ARM	X86
simSeconds	0.000376	0.000505
simTicks	375824000	505291000
finalTick	375824000	505291000
hostTickRate	9692230721	11636804554
hostMemory	632608	619280
simInsts	5708	6424
simOps	6523	11546
hostInstRate	146886	147685
hostOpRate	167825	265392
system.cpu.numCycles	375824	505291
system.cpu.exec_context.thread_0.numLoadInsts	1189	1140
system.cpu.exec_context.thread_0.numStoreInsts	975	954
system.cpu.exec_context.thread_0.statExecutedInstType::MemRead	1189	1140
system.cpu.exec_context.thread_0.statExecutedInstType::MemWrite	975	954
system.cpu.exec_context.thread_0.numIntRegReads	6655	16269
system.cpu.exec_context.thread_0.numIntRegWrites	3466	8963
system.cpu.exec_context.thread_0.numInsts	5708	6424
system.cpu.exec_context.thread_0.numOps	6523	11546
system.cpu.exec_context.thread_0.numCallsReturns	237	243
system.cpu.exec_context.thread_0.numCondCtrlInsts	920	1182

Part2

Things to keep in mind: ARM follows RISC ISA and X86 follows CISC ISA

Parameter	Reason for difference

Parameter	**Reason for difference**
simSeconds	It is lesser in ARM because it follows risc architecture which means that the instructions will execute using registers which allows risc architecture to execute more instructions in less time as registers are faster
simTicks	it is higher in X86 because it follows CISC architecture in which usually instructions may execute in more than one clock cycles so it requires more clock cycles to execute and hence the clock ticks are higher in cisc i.e. X86
finalTick	since finalTick stores the ticks from beginning without resetting so its reason for difference is similary to what have been given in simTicks
hostTickRate	it is higher in X86 because it follows CISC architecture in which usually instructions may execute in more than one clock cycles per second so it requires more clock cycles per second to execute and hence the host tick rate which is ticks per second are higher in cisc i.e. X86
hostMemory	it is lesser in X86 due to its cisc nature, since the code length is less in cisc so it would require lesser host memory as compared to ARM whihc follows risc
simInsts	X86 has more instructions simulated because of its cisc nature since cisc uses transisters to execute more instructions and also it has micro instructions as well which overall increase the number of instructions in cisc architecture
simOps	X86 has more operations simulated because in cisc there are instructions + microoperations are also there which makes number of operations in cisc on higher side
system.cpu.numCycles	number of cycles are higher in x86 because in cisc, instructions may execute in more than one cpu cycle as compared to risc (in which instruction execute in one cpu cycle)
system.cpu.exec_context.thread_0.numIntRegReads	it is higher in cisc case because cisc can read data from mem to reg, reg to reg, reg to mem so it has higher register reads

Parameter	Reason for difference
system.cpu.exec_context.thread_0.numIntRegWrites	it is higher in cisc case because cisc can write data from mem to reg, reg to reg, reg to mem so it has higher register reads

Part3

the major difference between fig1 and fig2 is that, in case of X86 architecture we have interrupts given a sample code from simple.py

```
if m5.defines.buildEnv['TARGET_ISA'] == "x86":
    system.cpu.interrupts[0].pio = system.membus.mem_side_ports
    system.cpu.interrupts[0].int_requestor = system.membus.cpu_side_ports
    system.cpu.interrupts[0].int_responder = system.membus.mem_side_ports
```

in the above code we can clearly see that it target_isa is x86 then it will connect the interrupts to the memory and these are directly connected to the memory bus and they are also not cached

rest other part is same for both ARM and X86 architecture i.e. from the code itself we can infer the figure by following the steps mentioned in the code i.e create timing cpu , create bus, connect ports etc.

Part4

Difference between (stats.txt) step 7 and 8 (X86)

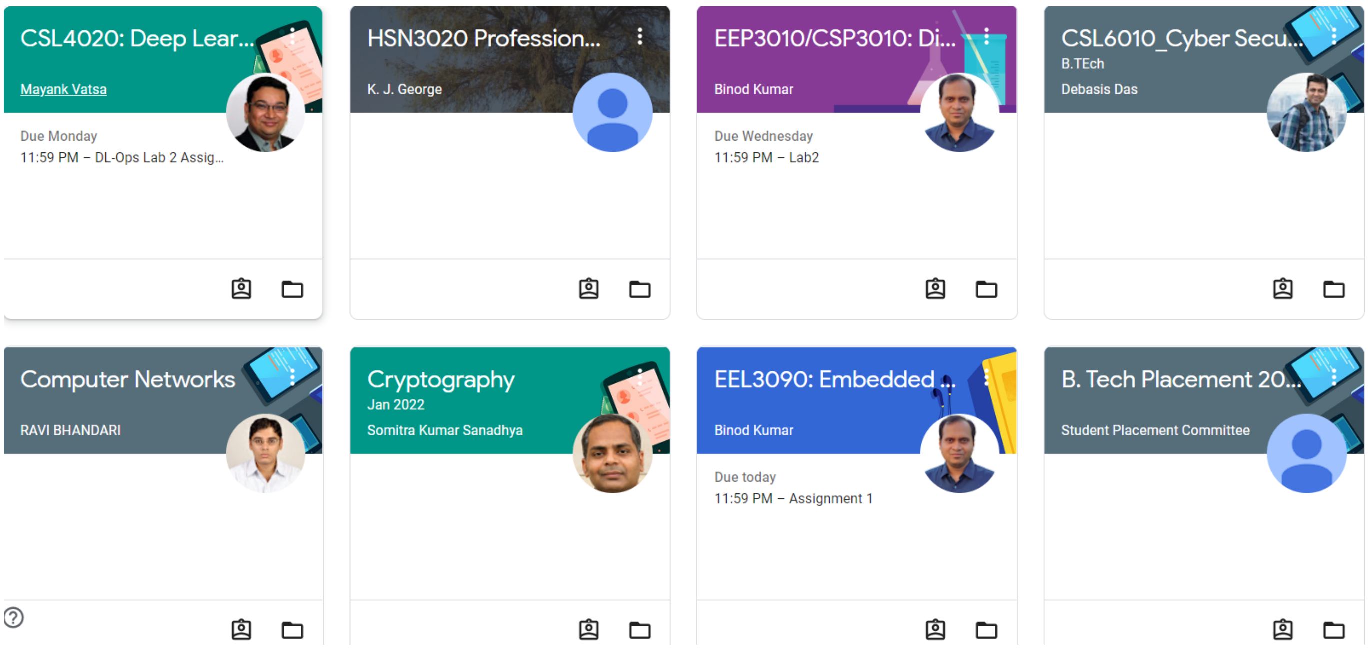
there are some visible differences among stats.txt file obtained from step7 and step8 some of them are as follows we can see that using se.py it simulates faster as compared to using simple.py and also it uses less ticks in case of se.py than simple.py but we can observe that se.py consumes more host memory than simple.py the number of cycles are also less in case of se.py than using simple.py

Difference between (stats.txt) step 6 and 9 (ARM)

there are some visible differences among stats.txt file obtained from step6 and step9 some of them are as follows we can see that using se.py it simulates faster as compared to using simple.py and also it uses less ticks in case of se.py than simple.py but we can observe that se.py consumes more host memory than simple.py the number of cycles are also less in case of se.py than using simple.py

Difference between simple.py and se.py

- se.py has memory managemant unit (mmu)
- se.py support multiprocess support
- se.py uses linux system calls
- se.py uses atomic simple cpu while simple.py uses timing simple cpu and atomic simple cpu is faster as it immediately finish memory request while timing simple cpu is slower as memory request takes time to go and return so that's why se.py is faster in simulation



CSL4020: Deep Lear...

Mayank Vatsa

Due Monday

11:59 PM – DL-Ops Lab 2 Assig...

HSN3020 Profession...

K. J. George

EEP3010/CSP3010: Di...

Binod Kumar

Due Wednesday

11:59 PM – Lab2

CSL6010_Cyber Secu...

B.TEch

Debasis Das

Computer Networks

RAVI BHANDARI

Cryptography

Jan 2022

Somitra Kumar Sanadhya

EEL3090: Embedded ...

Binod Kumar

Due today

11:59 PM – Assignment 1

B. Tech Placement 20...

Student Placement Committee