Indian Institute of Technology Tirupati
Computer System Architecture (CS5202)
Term Projects

[Max marks: 30][Due: April, 2020]

Instructions: You need to chose any of the projects from the given list. The list contains project of different nature. Each projects carry the same value in terms of marks. The choice is given such that best of your capability will be tested based on the outcome of the project.

---

1 Development and Design Project

1.1 Project 1

Problem: Development of a trace driven cache memory simulator
Group size: 2
Programming Language: C/C++/Java

Description: A cache memory simulator need to be designed as part of this project. It will be a trace driven simulator. The input to the simulator will be a trace file and a configuration file. The trace file contains a trace of physical address of each of instruction, next instruction, and an address of data. The format of trace file is provided in the Example 1, where the first address is the physical address of current instruction, next address is the physical address of next instruction, and the next field is data address if the data is located in memory or it is # if the data is non-memory type.

The simulator is expected to read from the trace file and perform the three primary function of cache placement, locate, and replacement. As part of the simulator design, it is expected to implement the desired algorithms for each of these functions. The simulator should be configurable by the user using the input configuration file. The configuration file contain the information about the essential parameters of cache architecture. Some of the essential parameters include cache memory size, block size, set size, mapping technique, replacement policies among others. As part of this project you are expected to design simulator for Level 1 cache. However, as bonus, you are encouraged to do the multilevel cache simulator.

Example 1: Trace and Configuration files

cache_trace_file.trc

```
//begining of file
// double slash for comment line
1000   1004 4001
1004   1008 #
.............
.............
//end of file
```

cache_configure.cfg

```
//beginig of file
// double slash for comment line
cache_size = 2 MB
block_size = 65 B
word_size  = 4 B
mapping    = 0 // mapping: direct map 0, set associative 1, fully associative 2
```

```
.............
.............
//end of file
```

Simulator Output: The simulator is suppose to generate sufficient number of statistics for the purpose of cache architecture study and research. Towards this the designer is expected to produce sufficient number of statistics on different parameters. The most important statistic being the total numer of cache reference, total miss, and total hit.

Project Outcome: A complete working simulator, a set of test case, implementation of one of the real life cache memories (example cache of i5, i7, or ARM A8).

1.2 Project 2

Problem: Implementation of superscalar scheduling algorithm (tomasulo algorithm)
Group size: 3
Programming Language: C/C++/Java

Description: The super-scalar architecture (to be precise it is a micro-architecture) is one of the instruction level parallelism methodology which build upon the pipeline architecture. One of the features of super-scalar architecture is out-of-order execution of instructions. To manage the out-of-order execution of instructions a dynamic scheduling algorithm is used which is originally called as Tomasulo's algorithm. The algorithm is being implemented in hardware using several components. The hardware design contain and manages the component such as reservation station, scheduler, reorder buffer, register file, load-store queue, and global data bus (common data bus). The important functionality of the algorithm is to take care of the data hazards: RAW, WAW, WAR, RAR. Unlike the traditional rigid in-order pipeline, for out-of-order super-scalar architecture all the three data hazards need to be handled: RAW, WAW, and WAR. The RAR does not affect the correctness of execution. For more detail you are advised to go through Chapter 3 of the text book.

In this project You need to design a trace driven simulator that simulate the behavior of Tomasulo's algorithm. The simulator need to capture the behavior of super-scalar processor from reservation station onwards to reorder buffer. As a bonus effort you may simulate the fetch and decode as well. However, it is not expected. It is very important to note that the simulator is a behavior simulator which works on the program trace as input. I have given you the format of trace file which captures necessary information about the program. The trace file contain physical address, instruction type (Integer ALU, Load, Store, and Branch), registers to read, a register to write. Based on these information (which we call as trace) the simulator suppose to schedule the instructions which are currently waiting in the reservation station to the execution units in out-of-order and then finish the execution in out-of-order with the help of reorder buffer. Finally, the register file has to be updated in program order. While doing all these functionality the important task is to check whether the registers to read is updated with the correct value or not. Note here that the simulator does not need to simulate the content of the register as such but the updated status of the register.

The simulator design should be such that it should be flexible enough to be configured using the configuration file. The configuration should be made in the sizes of each of the unit, number of execution units, sizes of common data bus, number of ports in the register file among others. It is strongly advised to study the Tomasulo's algorithm for much clarity. Innovation and addressing new problems in the existing algorithm could be considered as novel contribution of the project.

Simulator Output: The simulator must generate good number of statistics for architectural analysis.

Project Outcome: Working simulator with test cases. Preparing a trace file from the standard bench mark program would be added advantage.

1.3 Project 3

Problem: Study of Geometric Branch Predictor and Designing of Simulator
Group size: 2
Language: C/C++/Java

Description: Branch prediction has been one of the active research problems since last decade. In the pipeline architecture where the branch instruction get executed at the deeper stage of pipeline it is utmost necessary to know the program control flow. The advance knowledge of the program control flow helps in fetching the desired instruction and thereby save the pipeline clock cycles. However, knowing the program flow in advance is challenging task particularly for the conditional branch instructions. Therefore, the only way to know in advance is by prediction mechanism. The branch predictor simply predicts in advance, right at the instruction fetch stage using the BTB, the branch outcome, as either taken or not-taken. Based on this prediction of take or not-taken the next instruction is being fetched.

In this project the objective is to study the geometric branch predictor [**?**] using the help of simulator. The project expect innovation in terms of improving prediction accuracy, reducing hardware overhead and reducing dynamic power consumption. A trace based simulator need to be designed for the existing geometric predictor and the same is to extended for the novel predictor which you are expected to design. The simulator would take a traces of branch instructions as input and carry out the simulation accordingly. The trace file consists of physical address of the branch instruction, branch condition outcome, and the target address. The trace file format could differ for the different simulator design. The one which is provided here is designed by our team. However, you are free to use the different format as per your suitability. More trace file could be found in the Geometric paper [**?**].

The simulator must be configurable using the configuration file. The configuration should be as flexible as possible to allow the changes in history size, tag size among others.

Simulator Output: The simulator should provide the statistics on prediction accuracy. It should provide the finer detail such as prediction accuracy for each instruction and prediction accuracy of taken branch and not-taken branch.

Project Outcome: Fully working simulator, additional features to improve the prediction accuracy, trace file format, and configuration file. The test case trace file should resemble one of the real life program either from benchmark suit or from open source repository such as Sorting, Graphs etc.

trace_file.trc

```
//begining of file
//physical address  ----- Branch status    ----- Target Address
//of instruction    ----- after executn

AAAA000B                   0 (nottaken)     BB00000A
AACA000B                   1 (taken)        BB000C0B
ACCA000B                   0                BB000D0B
ABBA000B                   0                BB000E0B

//end ofile
```

configuration_file.cfg

```
//begining of file
//This is configuration file format, you could chose to have different format.
size_of_history = 1000 entry ;
base_predictor_size = 2 bit ;
tag_size = k bit ;
.................. ;
.................. ;
//end of file
```

1.4 Project 5

Problem: Smart Translation look aside buffer (S-TLB)
Group size: 2
Implementation: By use of simulator
Language: C/C++/Java

Description: The important functionality of TLB (translation look-aside buffer) is to map the virtual address with corresponding physical address. As the processor generate a request for instruction or data in the form of virtual address, the memory management unit ensure to locate the requested word in the main memory and serve it to the processor via memory hierarchy. All these functionality requires a translation of virtual address into physical address which is being done by TLB. Along with the adress translation the TLB also keep track of other information related to the page. Among the informations are read/write protection, user/supervisor mode, valid bit and others. The TLB is kind of cache memory. The memory management unit explicitly uses for address translation. The TLB usually has a fixed number of entry to it.

The objective of this project is to design a smart TLB which has adaptive feature in it. The motivation behind adaptive TLB design is to experiment on the appropriate size (in terms of number of entry) of the TLB. The design should be such that as needed the size should be shrinkable to less number of entry. Which means that if the TLB design has maximum of 40 entry and currently only 20 entry is being used and is sufficient for the current process, the remaining 20 entry should be turned off for power saving. And, whenever there is increase in TLB demand the remaining 20 entry should be made available for use.

The second motivation is to design a translation mechanism with more than one TLBs. There will be one TLB that is fixed for the purpose of address translation however the remaining will be used in shared mode. As per the need of program the remaining TLB can either be used data cache or can be used as TLB. The idea here is to improve the level 1 hit rate.

The experimental work in this project can be carried out in two ways: either by writing your own simulator (you can write a trace driven simulator) or by modifying the existing simulator such as CACTI. You may explore simplescalar simulator as well. If you are writing your own simulator, then decide on the trace file format and configuration file. To think about the trace file format, you can consider the page request that being generated by processor as virtual address.

Simulator output: There are two things to evaluate: first one is TLB hit and miss, the second one is how much power is being saved by turning off the part of TLB, and how much space is being appended to level 1 data cache. (To note: you need not implement data cache, what is expected is the space being saved.)

Project outcome: The project need to be demonstrated completely. If the project has been implemented with the existing simulator, then the portion of code that has been used for implementation need to be demonstrated. If the project has been implemented with your own simulator, it is expected that you need to design your own format for trace and configuration files.

1.5 Project 6

Problem: Cache coherence protocol and L3 cache
Group size: 2
Language: C/C++/Java

Description: The importance of cache coherence protocol comes when a multiple processor access a common location for the purpose of read and write. The ensure the consistent sharing of common data, it is essential to manage the read and write sequence among the participating processors using a set of rules. These set of rules form a coherence protocol. In the modern multi-processor architecture generally the L3 cache is shared among the multiple processors and therefore the cache coherence protocol is generally applicable to L3 cache. The protocol ensures the consistence communication among the multiple processors such that each processor would find the valid data when it access. This project

considers the centralized shared memory based multiprocessor architecture.

The objective of this project is to design a simulator for the L3 cache incorporating the snooping cache coherence protocol. Assuming an n number of processors sharing a L3 cache, the simulator should incorporate the way these processor need to communicate in order to pass the messages. The message passing protocol need to be in accordance with the snooping protocol. One of the important component of snooping protocol is the common bus system. Common bus is the medium through which the communication takes place. The simulator need to implement the common bus. While implementing the bus it is important to consider the contention kind of issues that may arise when more than one processor would like to get access to the bus. This needs to be arbitrated. As part of this project work it is expected to identify the limitations of the existing protocol and propose the new solution to it.

The simulation is based on the traces as input. Trace in this scenario could be the access request from various processors. Each processor would generate a series of access request to a common location at different point of time for either purpose of read or write. The simulator should be flexible enough to be able to simulate the behavior of at least 64 processors, and cache memory with varying size of block and total locations. For sake of simplicity you may consider direct map cache. Therefor, the simulator need to be configurable according to the configuration file.

Implementation Hints: The simulator design need to consider the state-transition machine. The protocol can be thought of as an state-transition machine where state represents the status of variables and transition represents the events such as next request, miss, and hit. The simulator also need to incorporate the clock tick to keep track of the event of interest.

Simulator output: The simulator should output the misses and their cause. How many time the buse have remain idle and used? When we say use it mean it is being used for transmitting and not just locking for future use.

Project outcome: Fully functional simulation with set of test cases. The test case should be able to test all possible events in terms of read and write.

1.6 Project 7

Problem: Data dependency and Basic block Visualizer
Group size = 2
Programming Language: C/C++/Java

Description: This project involves to solve the two related problem: data dependency and basic block identification. Design a to tool to find out the dependency among the instructions and draw an weighted graph showing the dependency relationship in the program. The edge weight depicting in the dependency graph depict the dependency distance and type of dependency where as the node depicts an instruction and its relative program counter value. The graph should be in format that would be visualizable by some existing tool.

The second part of the project involves designing of a similar tool that detects the exitence of basic block in a given assembly program. Given a assembly program, the task is to design a tool that would determine the basic block present in the program and design a visualization tool to visualise the relationship among the basic blocks using flow graph.

*The project would be further elaborated within two more days.

Tool output: The tool should output the dependency graphs and corresponding visualizer.

Project outcome: The expected outcome of the project would be a program that detects the dependencies by parsing the assembly code, the visualizer tool, and a set of test cases from the live program such as gcc, gdb etc.

Example:

Assembly program → parser → dependency/basicblock detector → graph generator → visualizer

1.7 Project 8

Problem: Main memory simulator for multi-bank system
Group size: 2
Language: C/C++/Java

Description: This project is to design a trace driven simulator for main memory multi-bank system. The simulator should incorporate the follow features of the main memory:

- Implementation of multibank system
- Implementation of little and big endian
- memory be tested using set of read and write operations.
- if memory is full, it should raise memory full error/page fault

The simulator would accept a trace file as input and a configuration file to configure the simulator to incorporate the features listed for varying sizes and number. More features better would be simulator statistics.
Simulator output: The simulator should provide the statistics on access time, sequence of byte address accessed, if there is any memory alignment problem.

* More information will be provided about the project by next two days.

1.8 Project 9

Problem: Optimal data structure of multi-level page table
Group size: 2
Language: C/C++

Description: Objective of this project to design an adaptive data structure (the word adaptive has been used here very loosely, it means something can be changed dynamically based on some events.) that would facilitate fast access to a page in the page table. Using that data structure, a page table simulator need to be designed. The simulator would work on taking input as trace of the page request from processor. Size of the page table entry, main memory size are important parameters among other. The simulator design should also consider the simulation of main memory, page placement and replacement as well (any policies of your choice could be implemented).

The adaptive data structure is something which is dynamically changeable as per the history of page request that comes from processor. The simulator designed should be done with both the standard data structure hierarchical tree structure and the new data structure.

*More description will be provided on this.

Simulator output: Statistics on number of memory access required to reach at the desired page, number of page fault, maximum level of page table, total size of memory occupied by page table among others.
Project outcome: A complete working simulator with test cases.

1.9 Project 10

Problem: In-order core with multiple threads
Group size: 2
Language: C/C++/Java

Description: The project is aimed at designing a simple in-order five stages pipeline simulator which executes multiple thread. The pipeline architecture will be single entry, however, for a multiple threads. When we say single entry it means from a particular thread the pipeline can fetch and execute only one instruction at a time, whereas, the design is free to fetch and execute from other thread. The project

could possibly be extended to incorporate the thread-level out-of-order execution. The simulator can be designed as trace driven by using trace file as input. The trace file should contain a sequence of instruction trace with necessary information as current physical address, next progam counter address, register to read, register to write, memory address to read and write.

Simulator output: The simulator is expected to generate a set of statistics on parameters such as number of clock cycle to execute the instruction, number halt cycles due to wait for data, for simply city you can consider unconditional branch only and assume that the jump address is available at fetch.

Project outcome: Working simulator with test cases.

*More description will be provided.

2 Analysis of the existing solutions/architecture

This is another set of projects you may chose to carryout. This set of projects are involved with sufficient number of experiments. You need to have a good computer to run the simulations. The idea here is to evaluate some of the methodology that has been proposed in the literature. The evaluation involves preparing a experimental set-up, implementing the methodology in simulation environment, and then performing appropriate experiments to collect the statistic on selected parameters on which the evaluation will be carried out. The evaluation should leads to the conclusion on pros and cons of the methodology or architecture under evaluation. This projects involves study of the paper, theoretical analysis of the idea, methodology, experimental set up, experimental results, and finally the pros and cons of the paper.

An example problem is suggested here for your reference.

2.1 Project 1

Problem: Performance per watt analysis of the Intel i7 processor
Group Size: 2
Description: The project work study the performance per watt of Intel i7 processor. The objective is to figure out the architectural and design shortcoming of the processor in providing the expected performance per watt. The outcome of the i7 will be compared with the respective ideal processor (theoretically best possible processor). The project will be carried out using the snipper simulator. The project involves following important task with respect to experimental work:

- Study the architecture and micro-architecture of i7 processor.
- Learn the usage of **sniper** simulator.
- Prepare a configuration file for i7 processor
- Run simulation and record the results

Along with the above experimental work, the architecture need to analysed theoretically for performance, power, and performance per watt.

3 Comparative study Project

This is another kind of project where a comparative study between more than one ideas (here idea means paper) can be carried out. The nature of the project work is very similar to the Evaluation project. However, this project involve comparative study of the papers. The paper chosen should be addressing exactly the same problems but solution proposed are of different kind. For example, cache performance improvement may be the problem of interest but solutions could be of different nature: one is software approach to deal with the never used cache block other could be hardware approach such as dead-block elimination. The comparative work involves the recreation of experimental set-up and implementation.

3.1 Project 1

Papers: The original works on value prediction

1. Gabbay, F and Mendelson, A. Speculative Execution Based on Value Prediction. Technion TR-1080, 1996.

2. Lipasti, M. H., and Shen, J. P. Exceeding the dataflow limit via value prediction. In Proc. Annual International Symposium on Microarchitecture (MICRO), 1996.

You are advised to discuss the project once you chose the papers to compare before you proceed for experiment, evaluation, comparison.

4 Solving existing problems

The computer architecture research have been primarily focused on the following three parameters: Power problem, Performance per watt (improving performance at a given temperature), and reducing hardware cost. When a specific component of the architecture being studied these three problems take their own specific definition. For example when we take branch prediction what it means for improving overall performance is reducing the miss prediction, optimizing the access time involved in BTB and other related design.

This projects requires you to study the research papers and find out some of the existing problems. Objective of the project is to **propose a new solution** which is expected to improve upon the existing solution by reasonable numbers. For example, if you are solving the performance issues due to cache memory, your solution should perform better by at least 5 - 10 % compared to the existing cache architecture. Some of the papers are listed here for reference reading.

- A. Jog, O. Kayiran, A. K. Mishra, M. T. Kandemir, O. Mutlu, R. Iyer, and C. R. Das, Orchestrated Scheduling and Prefetching for GPGPUs, In Proc. 40^{th} International Symposium on Computer Architecture (ISCA), Tel-Aviv, Israel, June 2013

- Aamer Jaleel, Joseph Nuzman, Adrian Moga, Simon C. Steely Jr., Joel Emer, High performing cache hierarchies for server workloads: Relaxing inclusion to capture the latency bene ts of exclusive caches, In Proc. of the 21st International Symposium on High Performance Computer Architecture (HPCA), Feb. 2015

- Manjunath Shevgoor, Sahil Koladiya, Rajeev Balasubramonian, Chris Wilkerson, Seth Pugsley, Zeshan Chishti, Efficiently Prefetching Complex Address Patterns, 48th ACM/IEEE Annnual International symposium on Microarchitecture (Micro 2015)

- M. K. Qureshi V. Srinivasan J. A. Rivers, Scalable High Performance Main Memory System Using Phase-Change Memory Technology, ISCA 2009.

5 Identifying new research problem

This problem is highly challenging, it involves extensive knowledge of computer architecture. The new problem could comes at different level of granularity. It has to be specific to some of the architectural detail such as execution unit, register file, cache memory etc. If you are interested in this project, kindly consult with instructor before proceeding further.

Some of the problem which are of current interest are listed here for further investigation:

- Application specific architecture to address power issues.

- Reducing the memory delay by introducing an intermediate level of memory using high speed flash drive.

Some of the views and future direction papers are listed here for your reference.

- Shekhar Borkar and Andrew Chien, 'The future of microprocessors', Communications of ACM, vol. 54, no. 5, May 2011
- Tilak Agerwala and S. Chatterjee, 'Computer architecture: challenges and opportunities for the next the decade', IEEE Micro, May-June 2005
- Mark Hill and Michael Marty, 'Amdahl's law in the multi-core era', IEEE Computers, July 2008
- Dong Hyuk Woo et al., 'Extending Amdahl's Law for Energy Efficient Computing in the Many Core Era', IEEE Computers, Dec 2008
- 21^{st} Century Computer Architecture, A community white paper, computing community consortium, May 2012
- Luis Ceze, Mark D Hill, T F Weinisch, 'Arch2030: A Vision of Computer Architecture Research over the Next 15 Years', Computing Community Consortium, 2016.

References

[1] Andre Seznec et al, A case for (partially) TAgged Geometric History Length Branch Predictor, AI Access Foundation and Morgan Kaufmaan Publishers, 2006