Undergraduate Computer Architecture, Fall 2020
Final Exam, 2020-01-11

Please take a look at the following items before you start to look at the questions.

- If you agree with the following sentence, please sign your name below it.

  I have not cheated nor have I received any help from other students in the exam.

  Student ID & name: B07905048 李宥霆

- Please return this sheet together with your answer sheet(s) when you complete the exam.
- Note that there are 110 points in this exam. Please try to get all the points that you can get.

1. (25 pts) Writing Parallel Processing Software

The difficulty with parallelism is not the hardware; it is that too few important application programs have been rewritten to complete tasks sooner on multiprocessors. It is difficult to write software that uses multiple processors to complete one task faster, and the problem gets worse as the number of processors increases.

(a) (15%) Why have parallel processing programs been so much harder to develop than sequential programs? Please explain the difficulties as complete as possible.

Hints: There may be several reasons, and the following keywords may be related to the answer: performance, efficiency, compiler optimization, automatic parallelization, load balance, task scheduling, interprocessor communication, synchronization, etc.

(b) (10%) Use an example code to illustrate 2 of difficulties mentioned in your previous answer and discuss how to solve the difficulties in the example code.

2. (25 pts) Performance and Availability

The small-form-factor hard disks for PCs in the mid-1980s led a group at Berkeley to propose *redundant arrays of inexpensive disks* (RAID). This group had worked on the reduced instruction set computer effort and so expected much faster processors to become available. Their two questions were: What could be done with the small disks that accompanied their PCs? What could be done in the area of I/O to keep up with much faster processors? They argued to *replace one large mainframe drive with 50 small drives, as you could get much greater performance with that many independent arms*. The many small drives even offered savings in power consumption and floor space.

(c) (5%) What does the paragraph mean by "greater performance with that many independent arms?" Please define the performance that can be improved by RAID, and explain why it can be done.

(d) (5%) While the idea of RAID can improve the performance and possibly save the power consumption and floor space, how does the action "replacing one large mainframe drive with 50 small drives" affects the *mean-time-to-failure* (MTTF) and availability?

(e) (5%) To improve the availability of the RAID system, fault-tolerant techniques can be used so that certain faults may not cause any failure. *Mirroring* is one of the simplest fault-tolerant techniques. One can use mirroring to duplicate the data from one disk to another disk. For example, in the above case, the 50 disks can be divided into 25 pairs, and each logical data block is stored in a pair of two disks. In this case, what would happen when one of the 50 disks fails? What should the administrator react to the failure of one disk? What would happen if another disk fails?

(f) (5%) Please discuss how this mirroring scheme in Question (e) would affect the MTTF? How does the resulted MTTF compare to that of a single disk?

(g) (5%) Today, many systems use solid-state disks (SSD) instead of traditional disks. Is the idea of RAID meaningful to SSD? Why?

3. **(20 pts) Virtual Machines**

Virtual machines (VM) have recently gained popularity due to several reasons, including:

- ✓ The increasing importance of isolation and security in modern systems
- ✓ The failures in security and reliability of standard operating systems
- ✓ The sharing of a single computer among many unrelated users, in particular for Cloud computing

(h) (5%) Some VMs can provide a complete system-level environment at the binary instruction set architecture (ISA) level. How may this type of VMs be used to improve security and reliability?

(i) (5%) Can a VM run a different ISA in the VM from the native hardware? Why would people do that?

(j) (5%) In the case of running a different ISA in the VM, it usually relies on a technique called binary translation, which is a form of binary recompilation where sequences of instructions are translated from a source instruction set to the target instruction set. Instead of having the virtual machine monitor (VMM) interprets one source instruction one-by-one during the runtime, it is possible to use a static binary translator to convert all of the binary code of an executable file into code that runs on the target architecture. What is the advantage of static translation over interpretation?

(k) (5%) Unfortunately, static translation can be very difficult to do correctly, since not all the code can be discovered by the translator. Instead of static translation, some translators use *dynamic binary translation* (DBT), which looks at a short sequence of code—typically on the order of a *single basic block*—then translates it and caches the resulting sequence. Code is only translated as it is discovered and when possible, and branch instructions are made to point to already translated and saved code. What does *basic block* means? Why does DBT looks at single basic block and cache the translated code?

4. (25 pts) Processor Design

The following table compares the ARM A53 processor to the Intel Core i7 920 processor.

Processor	ARM A53	Intel Core i7 920
Market	Personal Mobile Device	Server, Cloud
Thermal design power	100 milliWatts (1 core @ 1 GHz)	130 Watts
Clock rate	1.5 GHz	2.66 GHz
Cores/Chip	4 (configurable)	4
Floating point?	Yes	Yes
Multiple Issue?	Dynamic	Dynamic
Peak instructions/clock cycle	2	4
Pipeline Stages	8	14
Pipeline schedule	Static In-order	Dynamic Out-of-order with Speculation
Branch prediction	Hybrid	2-level
1st level caches/core	16-64 KiB I, 16-64 KiB D	32 KiB I, 32 KiB D
2nd level cache/core	128–2048 KiB (shared)	256 KiB (per core)
3rd level cache (shared)	(platform dependent)	2–8 MiB

(l) (10%) From the table, could you explain why the A53 processor consumes only 100 milliWatts with 1 core running at 1GHz, while the Intel i7 920 processor consumes far more power than that? Please list the top 3 factors and explain why they cause the differences.

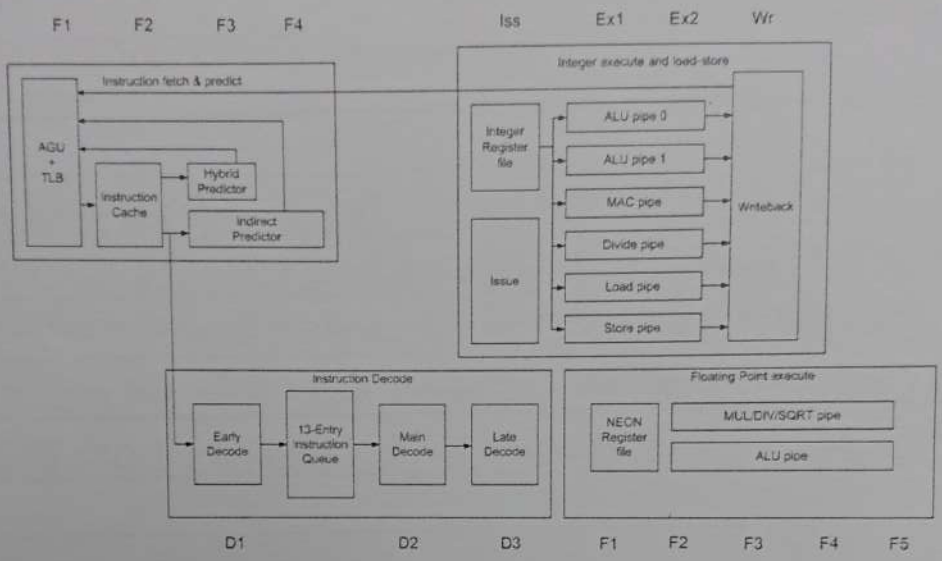
(m) (5%) Assuming we can put as many ARM A53 cores as we want on a chip, but the thermal design power is limited to 130W, how many cores on the chip can be turned on to run at 1 GHz? How does this chip compare to the Intel Core i7 920 processor, in terms of peak instruction throughput?

(n) (5%) Compared to the Intel Core i7 920, would you choose to put many A53 cores on a chip for personal computing? Why? Would you do it for server computing? Why?

(o) (5%) The table does not show an important architecture feature called TLB. Please explain the function of TLB and compare the TLB in these two processor designs.

5. (15 pts) Processor Pipeline and Performance

The following figure shows the pipeline structure for the ARM A53 mentioned in the previous question.



The first three stages fetch two instructions at a time and try to keep a 13-entry instruction queue full. The Address Generation Unit (AGU) uses a 6k-bit hybrid conditional branch predictor, a 256-entry indirect branch predictor, and an 8-entry return address stack to predict future function returns. The prediction of indirect branches takes an additional pipeline stage. This design choice will incur extra latency if the instruction queue cannot decouple the decode and execute stages from the fetch stage, primarily in the case of a branch misprediction or an instruction cache miss. When the branch prediction is wrong, it empties the pipeline, resulting in an eight-clock cycle misprediction penalty.

(p) (5%) Assuming the instruction cache and the data cache are ideal (100% hit) for our benchmark program, but the branch predictors combine to give 10% of misprediction rate, what would be the resulted CPI? 佔 10%

(q) (5%) Assume the branch prediction is perfect for our benchmark program, but the instruction cache and the data cache both yield 10% miss rate. If the miss penalty is 100 cycles for all misses, and the frequency of all loads and stores is 20%, what would be the resulted CPI?

(r) (5%) Now, suppose the pipeline of the A53 processor is changed to dynamic out-of-order execution with speculation, how would the change impact the CPI in the previous two questions?