Spring 2020 Final Exam Review

When / Where

Wednesday, 6 May 2020 to Friday, 8 May 2020 17:00 CDT.

Solve-Home, keep everyone safe in your bubble!

Conditions

Work on the exam alone.

E-mail questions to koppel@ece.lsu.edu. Don't be shy. Gotachas welcome.

You can use slides, references, and other materials provided for the course.

Do not seek out solutions to specific questions.

Format

Two or three or maybe four medium to long problems.

Short-answer questions.

Resources

Check course Web page daily for hints, new resources.

Web page: `https://www.ece.lsu.edu/ee4720/index.html`

RSS feed: `https://www.ece.lsu.edu/ee4720/rss_home.xml`

Solved tests and homework:. `https://www.ece.lsu.edu/ee4720/prev.html`

Study Recommendations

Study homework assigned this semester—**test questions are often based on homework questions.**

$\boxed{\text{\underline{\textit{Solve}} \textbf{previous test problems}}}$, start with more recent problems.

Memorizing solutions is not the same as solving problems.

Following and understanding solutions is not the same as solving problems.

Use the solutions for brief hints and to check your own solutions.

Emphases

Implementation Diagrams and Pipeline Execution Diagrams

They are a *team*, so study them together.

Designing and analyzing control logic and datapath.

Control Logic

We can't rely on magic.

Instruction Use

Should be able to easily write MIPS programs.

Should be able to use MIPS and SPARC instructions in examples.

Not required to memorize instruction names, except for common MIPS instructions.

Branch Predictors

Understand Operation, Determine Prediction Accuracy

Topics

Introductory Material

ISA v. Implementation.

Factors influencing ISA and implementation.

Benchmark types.

Compiling and Optimization

SPECcpu Benchmark Suite

See Lecture Slides 2—`https://www.ece.lsu.edu/ee4720/2020/lsli02.pdf`,
SPECcpu description—`http://www.spec.org/benchmarks.html`,
and the SPECcpu run and reporting rules—`http://www.spec.org/cpu2006/Docs/runrules.html`

What SPECcpu is supposed to measure. (Potential of new CPU implementations.)

Importance of having tester compile code.

Peak v. Base tuning: code preparation, use of results.

Why should we trust the SPEC rules?

Why should we trust the results of a test?

Benchmark programs (types, how they were selected).

Compilers and Optimization

Steps in building and compiling.

Basic optimization techniques, compiler optimization switches.

Profiling.

Compiler ISA and implementation switches.

How programmer typically uses compiler switches (options).

Control Transfer Instructions: Types, when to use.

Branch, Jump, Jump & Link, Call, Return

Format of displacements in instruction.

Specification of condition: condition code registers or integer registers.

Instruction Coding.

Fixed-length, variable-length, and bundled instructions.

Splitting of opcode field (as in MIPS type-R instructions).

ISA Classifications: RISC, CISC, VLIW

Dependency Definitions

Hazard Definitions

ISA Familiarity

MIPS

Read programs, write programs, implement (design hardware)

SPARC

Read common instructions.

Use of condition codes.

Instruction coding differences.

Register windows.

MIPS

Classification: RISC

Goals: ISA should allow simple, high-speed implementation.

Instruction types.

Know how to read and write MIPS programs.

Statically Scheduled MIPS Implementations

See Lecture Slides 6—`https://www.ece.lsu.edu/ee4720/2020/lsli06.pdf`
and Statically Scheduled Study Guide—`https://www.ece.lsu.edu/ee4720/guides/ssched.pdf`

Pipelined Implementations

Basic (no bypassing, 2-cycle branch penalty), bypassed, branch in ID.

For a Given Pipelined Implementation

Show pipeline execution diagrams.

Show register contents at any cycle.

Design control logic.

Add logic and datapath to provide new bypass, insn, etc.

Determine CPI.

Interrupts and Exceptions and Traps

Difference between interrupt, exception, trap.

Causes of exceptions, role of handler.

Privileged Mode.

Pipeline activity leading to execution of handler.

Precise exceptions, achieving with floating-point operations.

Long Latency (FP) Operations

Types of operations. (Floating point and maybe load.)

Degree of pipelining: Initiation interval.

Detecting functional unit structural hazards.

Detecting WB structural hazards: pipeline control logic.

Handling WAW hazards.

Superscalar and VLIW

n-Way Superscalar

Duplication of Resources.

Duplicated $n\times$: Fetch, decode, rename, writeback, commit.

Duplicated $< n\times$: load/store, floating-point units.

Costs: $\propto n$ functional units; $\propto n^2$ bypass, control.

Performance Limiters

Limiters due to device technology: Lower clock with increasing distances.

Aligned groups impose fetch restrictions that reduce fetch efficiency.

More stalls due to data dependencies.

More squashes due to branches.

VLIW

Difference with superscalar: instruction bundling.

Deeper Pipelining

Deeper (Super) Pipelining

Relationship between stage splitting, clock frequency and performance.

Relationship between stage splitting and cost.

Latch setup time.

More stalls due to data dependencies.

Vector Instructions

Vector registers.

How vector instructions operate on vector registers.

Advantages and disadvantages and differences . . .
. . . between vector instructions and superscalar systems.

Branch Prediction

Bimodal predictor.

Correlated branch predictors: local, global, gshare.

Branch target prediction.

Caches and Memory

Material not required for Spring 2020.